DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Managing the DG/UX™ System

093–701088–05

AVIION®
PRODUCT LINE

Managing the DG/UX™ System

093–701088–05

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

Printed in the United States of America
Rev. 05, July 1994
Licensed Material – Property of Data General Corporation
Ordering No. 093–701088

Notice

Managing the DG/UX™ System
093–701088–05

Revision History:		Effective with:
Original Release	– June, 1991	
Revision 1	– January, 1992	
Revision 2	– August, 1992	
Revision 3	– March, 1993	DG/UX System 5.4 Release 2.01
Revision 4	– January, 1994	DG/UX System 5.4 Release 3.00
Revision 5	– July, 1994	DG/UX System 5.4 Release 3.10

A vertical bar in the margin of a page indicates substantive technical change from the previous revision. See the Contents for organizational changes since the previous release.

Preface

This manual explains the concepts you need to understand and the tasks you perform to manage and customize the DG/UX™ system to meet the needs of your workplace.

See these other manuals for coverage of specialized management topics and tasks:

- *Achieving High Availability on AViiON® Systems*
- *Analyzing DG/UX™ System Performance*
- *Installing and Managing Printers on the DG/UX™ System*
- *Installing the DG/UX™ System*
- *Managing Mass Storage Devices and DG/UX™ File Systems*
- *Managing Modems and UUCP on the DG/UX™ System*
- *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*
- *Managing TCP/IP on the DG/UX™ System*

This manual assumes that you are familiar with computers but not necessarily familiar with the DG/UX system or other UNIX® operating systems. While detailed knowledge of the UNIX operating system is not required, it is helpful to know:

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

This manual covers these topics as they pertain to system management tasks. For more information on these topics, see *Using the DG/UX™ System*.

How this manual is organized

The manual contains the following chapters:

Chapter 1 **System management tasks**
Summarizes the jobs involved in managing a DG/UX system.

Chapter 2 **System management tools**
Explains the software tools you use to manage a

DG/UX system. Includes a description of the **sysadm** utility, Data General's menu-based system administration tool.

Chapter 3 **Booting, rebooting, and shutting down the system**
Explains how to boot and shut down the system and how to control the system processing involved in booting and shutting down the system.

Chapter 4 **DG/UX files, directories, and swap areas**
Describes the directory structure of DG/UX, how to work with files and directories, the contents of directories shipped with DG/UX, and how to control and reduce the size of automatically generated files.

Chapter 5 **Users, user groups, and user licenses**
Explains how to manage users and their login accounts, user groups, and user licenses.

Chapter 6 **Jobs, processes, and system activity**
Explains how to automate and schedule job execution, control processes, and monitor and control system activity.

Chapter 7 **Error logging, system dumps, and failure recovery**
Explains how to monitor system health, collect and view system and network messages and errors, detect and recover from system hangs, recover from system failures, create a system dump and diagnostic tape for STR submission, and avoid interruptions due to power failures.

Chapter 8 **Backing up and restoring files and file systems**
Explains how to back up and restore data.

Chapter 9 **Localizing the DG/UX system**
Explains how to set the system date and time, and how to tailor the DG/UX system for users based on regional and cultural specifics.

Chapter 10 **Terminals, ports, and tty lines**
Explains how to set up and manage terminals, ports, and tty lines for use with the DG/UX system.

Chapter 11 **Service Access Facility (SAF)**
Describes how to use the command line features of the Service Access Facility (SAF) to manage port monitors and services.

Chapter 12 **Controllers**
Explains how to set up and manage synchronous WAN controllers, LAN controllers, VDA controllers, and virtual terminal controllers (VTCs).

Chapter 13 **Loading and setting up application software**
Explains how to load and set up software other than the DG/UX system, such as the X Window System and TCP/IP.

Chapter 14 **Setting up secondary operating systems in software release areas**
Explains how to install a different release of the DG/UX system or another vendor's UNIX operating system on an OS server's disk for use by OS clients.

Chapter 15 **Building and booting the DG/UX kernel**
Explains when and how to build and boot a kernel.

Chapter 16 **Accounting**
Describes how to use the accounting system to monitor the use of system resources and generate and print summary reports.

Chapter 17 **Security**
Describes policies and tools you can use to enforce security on a standard DG/UX system.

Chapter 18 **OS servers and clients**
Explains how to set up and manage operating system (OS) servers and clients.

Chapter 19 **X terminal clients**
Explains how to set up and manage X terminal clients.

Chapter 20 **DG/UX device names**
Explains the device name format recognized by the DG/UX system and gives examples of valid device names.

Related manuals

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

Achieving High Availability on AViiON® Systems (093–701133). Describes the hardware and software components of

high-availability systems. Provides instructions for managing high-availability features of the DG/UX system. Explains how to set up AViiON systems in failover configurations, including examples of typical configurations.

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

Installing and Managing Printers on the DG/UX™ System (093-701132). Describes how to install, configure, and manage printers on the DG/UX system. It provides instructions for connecting cables, selecting the proper **stty** options and emulation modes, and troubleshooting printer problems.

Installing the DG/UX™ System (093-701087). Describes how to install the DG/UX operating system on AViiON hardware.

The Kornshell Command and Programming Language (093-701105). Provides a tutorial to the Kornshell command language and interface, explains how to create scripts using this language, and provides examples and reference pages.

Learning the UNIX® Operating System (069-701042). Helps beginners learn UNIX fundamentals through step-by-step tutorials. Published by O'Reilly and Associates and available from Data General.

Legato NetWorker Administrator's Guide (069-100495). This manual explains the setup and maintenance procedures for the NetWorker backup software, including the NetWorker server, its clients, and backup devices.

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

Managing Mass Storage Devices and DG/UX™ File Systems (093-701136). Explains how to manage disk and tape drives. Also explains DG/UX file systems, virtual disks, mirrors, and caching.

Managing Modems and UUCP on the DG/UX™ System (069-000698). Describes how to install and set up a modem that is compatible with the Hayes SmartModem™ in the DG/UX environment. Also describes how to set up and customize UUCP on the DG/UX system.

Managing ONC™ /NFS® and Its Facilities on the DG/UX™ System (093-701049). Explains how to manage and use the DG/UX

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

Managing TCP/IP on the DG/UX™ System (093-701051). Explains how to prepare for setup of Data General's TCP/IP (DG/UX) package on AViiON computer systems. Tells how to tailor the software for your site, and use **sysadm** to manage the package and troubleshoot system problems.

Programmer's Guide: STREAMS (UNIX System V Release 4) (093-701106). Describes the STREAMS interface facility and how to use it. The STREAMS facility provides special queuing, messaging and buffering functions that simplify addition and deletion of modules of code. For information on how STREAMS works in the DG/UX system and descriptions of important kernel-level utility routines, see *Programming in the DG/UX™ Kernel Environment* (093-701083). *Programmer's Guide: STREAMS (UNIX System V Release 4)* is published by Prentice Hall.

Using AViiON® Diagnostics and the AV/Alert℠ Diagnostic Support System (014-002183). Describes diagnostic products available for all AViiON computers. Provides steps for enabling and operating the AV/Alert system, and for using the utilities of the stand-alone AViiON System Diagnostics.

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™ System (069-701035). Describes the DG/UX system and its major features, including the C and Bourne shells, common user commands, the file system, and communications facilities such as **mailx**.

X Window System™ User's Guide, OSF/Motif™ Edition (069-100229). Describes window concepts, the application programs (clients) commonly distributed with the X Window system, and how you can expect programs to operate with the OSF/Motif interface.

Managing Security on the Trusted DG/UX™ System (093–701038). Describes security-related system administration tasks for the B1 and C2 versions of the Trusted DG/UX system.

Managing Auditing on the Trusted DG/UX™ System (93-701039). Describes administration tasks related to the audit system of the B1 and C2 versions of the Trusted DG/UX system.

Using Security Features on the Trusted DG/UX™ System (093-701037). Describes how to use security features provided with the B1 and C2 versions of the Trusted DG/UX system.

Format conventions

This manuals uses certain typefaces and symbols as follows.

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

Contacting Data General

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

Manuals

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

Telephone assistance

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

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

Joining our users group

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

End of Preface

Contents

Chapter 1 – System management tasks

Chapter 2 – System management tools

Chapter 3 – Booting, rebooting, and shutting down the system

Chapter 4 – DG/UX files, directories, and swap areas

Chapter 5 – Users, user groups, and user licenses

Chapter 6 – Jobs, processes, and system activity

Chapter 7 – Error logging, system dumps, and failure recovery

Chapter 10 – Terminals, ports, and tty lines

Chapter 11 – Service Access Facility (SAF)

Chapter 12 – Controllers

Chapter 13 – Loading and setting up application software

Chapter 14 – Setting up secondary operating systems in software release areas

Chapter 15 – Building and rebooting a DG/UX kernel

Chapter 16 – Accounting

Chapter 17 – Security

Chapter 18 – OS servers and clients

Chapter 19 – X terminal clients

Chapter 20 – DG/UX device names

Glossary

Tables

Table

Figures

Figure

1 System management tasks

This chapter introduces you to basic tasks you perform to manage the DG/UX system.

Installing the DG/UX system

One of the first jobs you may have is installing the DG/UX system on your computers. For how to install the DG/UX system, see *Installing the DG/UX™ System.*

Communicating with users

Try to provide as much advance notice as possible to users about events affecting the use of the system. When you must take the system out of service, be sure to tell users when the system will be available again.

You may want to keep the following guidelines in mind before performing any task that will require the system to leave the multi-user state.

- When possible, schedule system maintenance during periods of low system use.
- See if anyone is logged in before taking any actions that affect users. Always give users enough time (at least 60 seconds) to finish whatever they are doing and log off before taking stand-alone or server systems down.

For how to use utilities available for communicating with users to inform and alert them about system developments, see Chapter 2.

Maintaining a system log

We recommend that you maintain a detailed system log, both on paper and on disk, of the status of your system. This log may contain the following information:

- What devices are configured into the current kernel. Configured devices are listed in the system file which is located at **/usr/src/uts/aviion/Build/system.***name*, where *name* is usually the host's name.
- Equipment and system configuration changes (dates and actions)

- A record of any system panics and hangs and activities occurring at the time of failure
- Maintenance records (dates and actions)
- A record of problems and solutions

Whatever format you choose for your log, make sure that the system log and the items noted there follow a logical structure. The way you use your system will dictate the form that the logs takes and the diligence with which you maintain it. 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.

For problems you encounter after attempting failure recovery, we request that you fill out a Software Trouble Report (STR) and deliver it to the Data General Customer Support Center. For more information on STRs, see the DG/UX system release notice, shipped on-line in the **/usr/release/dgux_5.4R3.10.rn** file.

Maintaining a problem log

You may find it useful to log problems experienced by your users and their solutions in a problem log. With a well-maintained log, you can not only keep track of the solutions to common problems, you may also be able to detect patterns in the problems you encounter, possibly indicating ways in which you can improve service for your users.

Managing mass storage devices and DG/UX file systems

You must set up and maintain mass storage devices such as disk drives and tape drives to give users access to data. On drives and other mass storage devices, you set up DG/UX file systems to give both your users and the DG/UX system access to data.

For information on how to set up and manage mass storage devices and DG/UX file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Managing modems and UUCP

If users of your system have a need to communicate using modems and the UUCP product, see *Managing Modems and UUCP on the DG/UX™ System*.

Managing printers

You must set up printers and make them accessible by the DG/UX system for users to print file contents. For information about printers, see *Installing and Managing Printers on the DG/UX™ System*.

Monitoring system activity

The DG/UX system provides utilities that let you gather and review statistics on CPU performance, disk and terminal I/O, memory usage, process communication and execution, paging and memory usage statistics, and other system activity. When the system appears to be functioning erratically, or simply as a matter of course, you may want to review the system activity data.

For information on and how to use these utilities, see Chapter 6. For how to use these utilities to assist you in improving system performance, see *Analyzing DG/UX™ System Performance*.

Monitoring and controlling processes

A process is any program currently running on the system. The term refers not only to the executing program code but also to the program's environment and state for that instance of execution.

There will be times when you want to cancel or kill a process, or change the priority of the process to lessen or increase the amount of computer time it consumes. For how to monitor and control process activity, see Chapter 6.

Handling system errors and messages

Various system facilities produce messages during normal operation and when they encounter errors and other unexpected conditions. **Sysadm** automates a procedure for using the system error log. See Chapter 7 for more information.

Monitoring mail for administrative IDs

Some system services send mail to the **root** user about urgent or abnormal events. Your **root** mail file may also receive messages addressed to **postmaster**, **sysadm**, or another administrative title at your site.

If you log in as a normal user and use **su** to become the superuser as needed, you may not see the **root** mail notification message that

appears at login. Therefore, you should check the mail for **root** occasionally, or add a line to one of your personal configuration files (such as **.profile**, **.login**, or **.cshrc**) that checks for **root** mail when you log in.

Automating job execution

You may find it helpful to schedule jobs to run on a regular basis, or at times when the demand for the system is low. For example, you may have a script that checks disk free space and file system security. With the **cron** facility, you can schedule this script to run once a week or every night, for instance. With the **at** and **batch** facilities, you can schedule jobs at low priority or at any time you specify.

For how to use the **cron**, **at**, and **batch** facilities to schedule jobs, see Chapter 6.

Improving performance

There are different strategies and software tools you can use to analyze and improve the performance of your system. For example, you can change the values of some system parameters to handle your system load better and improve performance. Also, if you run applications with special performance needs, you can experiment with different combinations of parameter values to find an optimal set to support those needs.

For descriptions of tunable system parameters, see the **/usr/etc/master.d/dgux** file and *Analyzing DG/UX™ System Performance*. For how to use the system parameters to control system performance, and for explanations of strategies and tools you can use to improve the performance of your system, see *Analyzing DG/UX™ System Performance*.

Achieving high availability

High availability is the state of your computer operations when you can provide users with near-continuous availability of system resources and services. No single system component can provide high availability. However, you can maximize system availability and minimize downtime by strategic integration of hardware and software resources and services.

For a detailed explanation of high availability and the solutions Data General provides to help you achieve it, see *Achieving High Availability on AViiON® Systems*.

Setting the system time

The physical location of your computer operation determines the time you should set on your system. For how to set the time on your system, see Chapter 9.

Setting native language characteristics

The location of your computer operation and the linguistical and cultural background of your users determine the types of input the system should accept and the style of system's output. For example, certain DG/UX commands can accept native language responses to yes or no questions. Customizing your system to understand the characteristics of a French speaking location, for example, enables these commands to accept **oui** and **o** as valid responses in addition to the English language responses **yes** and **y**.

When you customize the DG/UX system to function within the geographical and linguistical context your users expect, you are localizing the system. In addition to allowing recognition of native language responses, localizing the DG/UX system can enable the display and recognition of culturally recognized currency symbols, the performance of sort operations using different collating sequences, and the culturally appropriate display of commas and decimal points in numerals.

For a complete discussion about localizing the DG/UX system and how to do it, see Chapter 9.

Backing up files and file systems

For how to back up files and file systems, and how to restore backups, see Chapter 8.

Maintaining and verifying system security

The DG/UX system provides a number of security features. If your operating environment requires a high degree of security, you may want to use the Trusted DG/UX systems that provide B1 and C2 levels of security.

For how to maintain a secure DG/UX system, see Chapter 17. For information about the Trusted DG/UX systems, contact your Data General representative.

End of Chapter

2 System management tools

This chapter discusses software tools you can use to manage the DG/UX system and how to use them. It also provides brief explanations of the AViiON® computer configurations you may manage.

For a complete discussion of the DG/UX system's user environment, including complete coverage of the shells, user commands, and editors, see *Using the DG/UX™ System* and *Using the DG/UX™ System Editors*. There is also a discussion of the **vi** editor in Chapter 4.

Logging in to the DG/UX system

You can log in to the system when the login prompt is on the screen. The login prompt you see depends on whether you're using a graphics monitor or an alphanumeric display terminal.

Logging in with a graphics monitor

If your system console is a graphics monitor and you installed the X Window System package, you see the following login box with four control buttons displayed on the screen (see Figure 2–1).

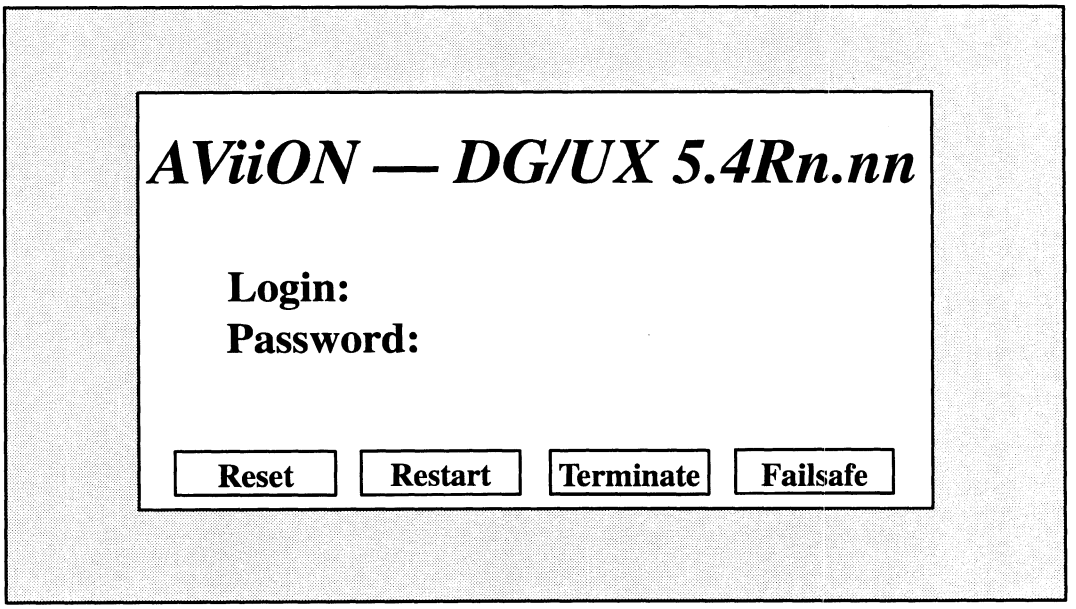


Figure 2–1 Login screen for a graphics monitor

On your screen, the *n.nn* is replaced by the number of the DG/UX release running on the system; for example, 3.10 if DG/UX 5.4 Release 3.10 is running.

The four buttons along the bottom of the screen are aids to controlling the X server, the program that controls the display of information:

Reset Resets the X server. The X server is not terminated.

Restart Terminates the X server, then restarts it.

Terminate Terminates the X server, giving control to a single VT100 terminal emulation screen.

Failsafe Is a toggle button limiting startup to a single Xterm window.

IMPORTANT: If you choose to exit the X Window environment and continue the procedures in a single VT100 terminal screen environment, move the cursor to the "Terminate" button and click. To return to the X Window environment, enter **xdm** at the login prompt. The X Window environment will be restored.

A single window and a login prompt appear.

Logging in with an alphanumeric display terminal

Figure 2–2 shows a login display for an alphanumeric terminal, with superuser **sysadm** logging into system **bob**.

```
bob
DG/UX 5.4R3.00 AViiON
Console login: sysadm
Password:
Copyright (c) Data General Corporation, 1984-1994
All Rights Reserved

============================================================
#                                                          #
#                        WARNING                           #
#                                                          #
# ACCESS TO AND USE OF THIS SYSTEM IS RESTRICTED TO        #
#               AUTHORIZED INDIVIDUALS                     #
#                                                          #
#          Data General AViiON DG/UX System                #
#                                                          #
============================================================

#
```

Figure 2–2 Alphanumeric display terminal as a system console

In this alphanumeric display example, **bob** is the nodename you specified during TCP/IP setup. The login banner appears, followed by the superuser prompt (#). In this case, a password was not needed to login.

The superuser and administrative logins

The DG/UX system restricts certain administrative commands and operations to the superuser. The superuser can execute, open, read, delete, or change any file in the system. Two default superuser logins, **root** and **sysadm**, exist on all DG/UX systems and can override any file permission settings. These two logins are also considered administrative logins.

To become the **sysadm** or **root** superuser, you can log in using one of these logins. If you are already logged in with another login, you can become the superuser by executing the **su** command with the - (hyphen) option followed by the desired login name. Either way, you must know the password for a superuser login to become that superuser.

To become **sysadm** after login, enter:

```
% su - sysadm ↵
```

The **su** command prompts you for the **sysadm** password. After you enter the password and become the **sysadm** superuser, you see the superuser prompt, #, instead of one of the user prompts (% or $).

IMPORTANT: You should assign passwords to the **sysadm** and **root** superusers. If these superuser logins have no passwords, anyone can log in to your system with the pervasive privileges of these logins and pose a major threat to the security and stability of your system.

You should use the **sysadm** rather than the **root** login when performing administrative tasks, so that commands executed can be performed and output generated in the **/admin** directory rather than the / directory, the home directory of the **root** login.

The DG/UX system can convey system file ownership among 12 login names. Five of these login names function normally; that is, you can become these with **su**. The other seven are for system use only; that is, you never actually log in with them. If you look at the **/etc/passwd** file, you'll see that the system logins have an asterisk (*) in the password field, meaning that no one can log in with these.

Table 2–1 shows the administrative and system logins.

Table 2–1 Default DG/UX 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**.
xdm	Your system has this login only if you have installed the X11 package. Logging in as **xdm** executes **telxdm**, the **xdm** control utility. See the **telxdm** manual page for more information. **xdm** is the X Window System session manager for systems with graphics capabilities.
sysadm	Same as **root**, except the login directory is **/admin**. You should use this login for performing administrative tasks.
daemon *	This is the login of the system daemon, which controls background processing.
bin *	This login owns the files in **/usr/bin**.
sys *	This login owns the files in **/usr/src**.
adm	This login owns the files in **/var/adm**.
uucp *	This login owns the object and spooled data files in **/usr/lib/uucp** and **/etc/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	You can log in to the system as **nuucp** and perform general administrative tasks. Some UUCP facilities may send messages to **nuucp**'s mail file (**/var/mail/nuucp** by default).
lp *	This login owns the object and spooled data files in **/var/spool/lp**, **/etc/lp**, and **/usr/lib/lp**.
mail *	This is the login of the electronic mail facilities. Some system facilities may send messages to **mail**'s mail file (**/var/mail/mail** by default).
sync *	Logging in as **sync** causes the system to execute the **sync**(1M) command before returning you to the login prompt. The **sync** command no longer performs any function; therefore, logging in as **sync** has no effect at all. The **sync** command and the **sync** login exist only for compatibility with previous revisions of the system.

In Table 2–1, only those entries without asterisks can be used as actual logins; the others are for system use only. Additional software packages may add other logins to the **/etc/passwd** file.

Recovering forgotten superuser passwords

If you forget both of the superuser (**root** and **sysadm**) passwords, follow the steps in this section to set a new one. You may be in either of two situations when you realize that you have forgotten the superuser passwords:

- You are logged on as **root** or **sysadm**, and you have the # prompt. Simply run the **passwd** command and set a new **root** and/or **sysadm** password.
- You are not currently logged on as **root** or **sysadm**. For instance, you may be logged on as yourself and have the normal shell prompt. Because there is no way to access the **root** or **sysadm** logins, 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 the wall command to warn users that the system is about to go down.**

   Wait until users have logged off or have at least terminated any processes (such as editors) that write to disk files.

2. **Wait 60 seconds before resetting 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 square brackets ( ] [ ] [ ] [ ) pressed while you hold down the control key:

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

   Resetting the system takes you to the SCM prompt.

3. **Reboot your system to bring it up to single-user mode.**

   If you have configured your system to come up to a run level other than single-user mode, you need to specify the **-S** option on the SCM **boot** command line. For example:

   ```
   SCM> b sd(cisc(),0)root:/dgux -S ↵
   ```

4. **If the attempt to boot fails because the root file system is corrupt, boot stand-alone sysadm, either from disk or from the release tape, and check the root file system.**

   If you need to repair damaged DG/UX system files, see Chapter 7.

5. **Once you are in single-user mode, assign new root and sysadm passwords with the passwd command.**

Tools for communicating with users

This section discusses several of the utilities you can use to keep your users informed of system news and administrative developments.

Message of the day (motd)

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** and **ksh** 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 to publicize down-time:

```
5/30: The system will be down from 6-11pm Thursday
for preventive maintenance. Call Bob if you have a
problem with this.
```

News

The **news** facility, an electronic bulletin board, provides a convenient means of distributing announcements to users. The facility maintains a directory, **/usr/news**, where you can put 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.

You can also use the **/etc/profile** or **/etc/login.csh** file to 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 **/etc/motd**, 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 wants simply to eliminate the display of item names without looking at the items, a couple of techniques will work:

- Use the **touch**(1) command to change the time-accessed and time-modified of the **.news_time** file, thus causing **news** to consider all existing news announcements as having already been read. The following example shows how to use the **touch** command for this purpose:

```
$ touch .news_time ↵
```

- Invoke **news** to read the articles, but direct the output to **/dev/null**. Like the previous technique, this one causes **news** to update the time stamp of the **.new_time** file:

```
$ news > /dev/null ↵
```

Broadcast to all users

With the **wall** command, you can 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 log out of the system so that you may perform an administrative task. For example:

```
% wall ↵
System going down at 3:00 for oil change -- Marv ↵
<Ctrl-D>
```

In network environments, there is also the **rwall** command. Use the **rwall** command to send a message to all users on the specified systems. For example:

```
% rwall sales03 accounts lifers ↵
The new desk blotters are in!  Bobbi.
<Ctrl-D>
```

DG/UX system mail

The DG/UX system has three electronic mail utilities that users can use to communicate among themselves: **mail**, **mailx**, and **xdgmail**. If your system is connected to others by networking facilities, you can use these utilities 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. The **xdgmail** program, which also provides a number of extra features, is for workstation users in an OSF/Motif™ environment.

Users can, by default, send and receive mail when you add them to the system with the Add operation. A simple example of using **mailx** follows.

Assume that Poulet wants to send a mail message to Moe. He types:

```
$ mailx moe ↵
Subject: rubber chicken ↵
Please return my chicken when the puppet show is over. ↵
<Ctrl-D>
```

Poulet enters a subject line when prompted, presses Enter, 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 Enter key, or when he logs in, his screen will display:

```
You have new mail.
```

Then Moe invokes his mail utility of choice to see what mail has arrived.

A setup file called **.mailrc** contains the mail characteristics for each user. For information on the **.mailrc** file, see the **mailx** manual page. For detailed information on using mail facilities, see *Using the DG/UX™ System* and the **mail**, **mailx**, and **xdgmail** manual pages.

Mail aliases

A mail alias is a name that stands for one or more user login names or mail aliases. You can set up mail aliases to make communicate more efficient between you and particular groups of users.

Mail aliases are useful for two primary purposes:

- A mail alias that stands for a list of user login names can act as a mailing list so that any mail sent to the alias name goes out to all names in the alias. For example, you can have an alias called **muleskinners** that resolves to login names **willie**, **eldupree**, **paco**, and **jed**.
- A mail alias that stands for one name can provide an easy way of redirecting mail sent to a general-purpose address. For example, you can set up the alias **wizard** so it directs mail to the resident computer expert at your facility. If the resident expert gets tired of answering user questions or simply runs out of answers altogether, you can change the **wizard** alias to point to another system expert. Users sending mail to **wizard** will always reach the resident expert, no matter who fills that job.

A default mail alias, **everybody**, is provided with the DG/UX system. Sending mail to the **everybody** mail alias sends mail to every user known to your system.

The Mail Alias menu of the **sysadm** User menu provides operations for adding, deleting, modifying, and listing mail aliases. You must be superuser to add, delete, or modify mail aliases. The system keeps track of mail aliases in the **/etc/aliases** file. The master server in an NIS domain also tracks mail aliases.

The **sendmail**-based mail facility and the **mailx** command can recognize and use mail aliases.

The ability of a system to recognize mailing aliases changes if the system is part of a network using the Network Information Service (NIS) facility (formerly called Yellow Pages or YP). For information on how the NIS facility affects a system, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

Adding mail aliases

Select the Add operation to add a mail alias to your system. The operation adds the mail alias to your local **/etc/aliases** file unless your system is the master server of an NIS domain, in which case you may choose whether to add the entry to the local **aliases** file or to the global NIS database.

The Add operation first prompts you to supply an alias name. The name can be up to 32 characters long, consisting of upper- and lowercase letters, numerals, and the hyphen (-). The first character should be an alphabetic character. The name must be unique among aliases on your system and, if applicable, in your NIS domain.

The operation also prompts you for an optional list of names. The names should be existing login names or existing alias names. You can include a login name or a mail alias in multiple mail aliases.

The new alias becomes available in the mail system immediately if you are changing the local **aliases** file. If you are changing the global NIS database, the alias may not be available until the following day.

The Add operation uses the **newaliases**(1C) command to make the new alias available.

Displaying mail aliases

Select the List operation to display mail aliases. If your system is in an NIS domain, you may choose whether you want to display aliases from the local **/etc/aliases** file or from the global NIS database.

An example of an alias listing follows.

```
Mail Alias              Mail Address(es)
----------              ----------------
MAILER-DAEMON           root

postmaster              root

aunts                   polly
                        jemima
                        grizelda

cowpokes                bubba@blarney.state.edu
                        sales03!wilbur@acme.com
                        george@grumble.mil
```

Modifying mail aliases

To change the name of an alias or to change the list of names to which an alias resolves, select the Modify operation. The operation changes the local **/etc/aliases** file unless your system is the master server of an NIS domain, in which case the operation lets you choose whether to change the local **aliases** file or the global NIS database.

The operation first prompts you for the name of the alias you wish to change. The alias must already exist. The operation also allows you to list the alias before entering a change.

For changing the list of names in the alias, the next prompt, **Member Modification**, offers four choices:

no change
Select this value if you do not wish to change the alias list.

append
Select this value to add names to the alias list. At the next prompt, **Alias Member(s)**, specify the names to append to the current alias list.

remove
Select this value to delete names from the alias. At the next prompt, **Alias Member(s)**, specify the names to remove from the alias list.

replace all
Select this value if you wish to replace the entire alias list with a new list. At the next prompt, **Alias Member(s)**, specify the new list of alias names.

At the prompt, **New Alias Name**, you may specify a new name for the alias if you wish.

Changes become visible in the mail environment immediately. If you change the global NIS database, changes may not be visible until the following day.

Deleting mail aliases

Select the Delete operation to remove an alias from the alias database. The operation removes the alias from the local **/etc/aliases** file unless your system is the master server of an NIS domain, in which case you may choose whether to remove the alias from the local **aliases** file or from the global NIS database.

Once you have deleted a mail alias, the mail system considers mail addressed to the alias to be misdirected mail. The mail system then returns the mailed message to the sender or forwards the mail to the **root** mail file.

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 ↵
```

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.

On-line documentation for commands, system calls, and files

A manual page, also called a man page, is an on-line document containing a complete description of each command, system call, and special file available on the DG/UX system. These commands, system calls, and files are collectively referred to as *utilities*.

A man page is a common convention used by the developers of the UNIX system to document their system within the system itself.

This manual makes frequent references to particular man pages that you can consult for more information. You use the **man** command to access these on-line man pages.

Viewing a man page

For example, to see the man page for the **man** command, you type the following:

```
$ man man ↵
```

Many man pages are too long to fit on one screen so they scroll quickly up and off the screen. To insert a pause between screens, you can use the **more** command along with the **man** command to control the output display. For example:

```
$ man man | more -f ↵
```

IMPORTANT: The vertical bar is referred to as a pipe. For more information on pipes, see the manual *Using the DG/UX™ System*.

At the bottom of the first screen, you will see this message displayed:

```
-- More --
```

Press the space bar to display the next screen of the man page.

Printing a man page

To print a man page, use this command format:

man –Tlp [*single–digit*] *manpage-name* **| lp –S iso–88591 -d** *printer-name*

where:

–Tlp
specifies a printer as a terminal type.

[*single–digit*] *manpage-name*
specifies an optional digit (**0–9**) corresponding to the desired type of man page. See Table 2–2 for man page types.

–lp
is the print command.

–S iso–88591
specifies the character set in which the set of man pages are formatted.

–d *printer–name*
identifies the printer's name.

For example, to print the **more** man page to a printer named **laser**, enter:

```
$ man -Tlp 1 more | lp -S iso-88591 -d laser ↵
```

Finding the man pages on a particular topic

To use the **man** command to find information, you must know the name of the command or file or system call you want. However, if you do not know specific names, you can use the **apropos** command to generate a list of man pages relevant to the topic about which you need information. The list includes a brief explanation of each listed man page. Then you can use the **man** command to look at the listed pages.

For example, assume you want to know what man pages are relevant to terminals. You can enter:

```
apropos terminal
```

The list generated by the **apropos** command will include lines like the following one:

```
getty(1M):   getty - set terminal type, modes, speed, and line discipline
```

For more information on how to use the **apropos** command, see the **apropos** man page.

Man page standard format and categories

At the top of each man page is the command name, such as **man**. Each man page is formatted to contain the following categories of information, where appropriate:

NAME	Names and briefly describes the utility.
SYNOPSIS	Gives utility syntax; the command name is presented in boldface type, and its options are presented in regular type; optional arguments are enclosed in brackets. An ellipsis (...) following an argument indicates you can repeat it.
DESCRIPTION	Gives a more detailed account of the utility and its arguments.
EXAMPLE	Illustrates the use of a command or concept.
FILES	Lists any system files used by the utility.
DIAGNOSTICS	Gives possible error messages and recovery actions.
SEE ALSO	Lists any related utilities.
BUGS/CAVEATS	Informs you of any known programming and design bugs or limitations.

Each documented command, system call, and special file is associated with a number signifying its type. In print and on-line, this number may appear in parentheses alongside the utility name. Table 2–2 lists the types of entries.

Table 2–2 man page types

Number	Class
0	Table of contents and permuted keyword–in–context index.
1	Commands and application programs
2	System calls
3	Subroutines and libraries
4	File format
5	Miscellaneous
6	Communications protocols
7	System special files
8	System maintenance procedures

On-line books

This book and other DG/UX user, administrator, and programmer manuals are provided by Data General on CD-ROM.

If you or your users need to refer to DG/UX manuals, accessing the manuals on-line can help you find information quickly. For example, the viewing software that comes with the on-line books includes a full-text search feature; this feature lets you locate information about a particular topic in several books within seconds.

For more information on the CD-ROM version of the DG/UX system documentation and how to order it, contact Data General or your Data General sales representative.

The system administration utility: sysadm

You can use the **sysadm** utility, a menu-based interface, to manage your system. Three interfaces are available:

- Graphical
- ASCII terminal menu
- Shell command line

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

Regardless of the interface you're using, you can type **sysadm** at the superuser prompt (#) to automatically start the interface that will run on your display:

```
# sysadm ↵
```

The system invokes the appropriate interface — the ASCII version (**asysadm**) or graphical version (**xsysadm**) of the **sysadm** utility — based on your display type. If you are working in an X windowing environment, invocation of **sysadm** will produce an X version of **sysadm**; when working in an ASCII terminal environment, by default, you will get an ASCII version of **sysadm**. You may choose explicitly the desired interface, however, by invoking either **xsysadm** or **asysadm**. The graphical and ASCII interfaces are functionally equivalent.

If you are using a graphical computer system such as a workstation or X terminal, you can type **xsysadm** for the graphical interface. Figure 2–3 shows the **sysadm** main menu as it appears in the graphical interface.

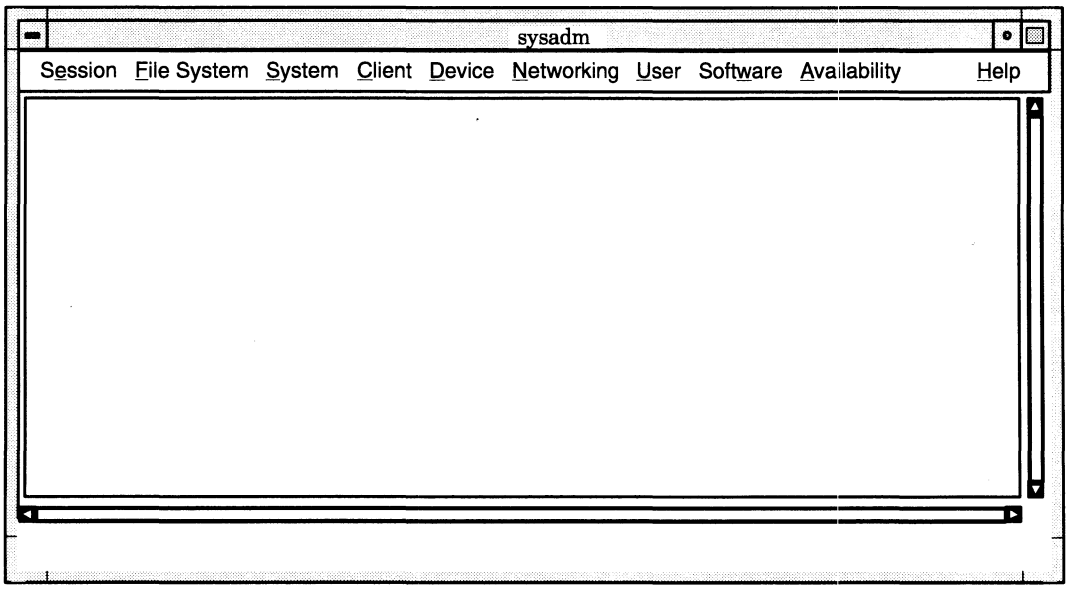


Figure 2–3 Graphical sysadm main menu

The window that actually appears on your system may have more or fewer selections than those shown in Figure 2–3, because the **sysadm** window and menu displays vary depending on the software loaded on the system.

If you are working from an ASCII termina, or if you enter **asysadm** instead of **sysadm**, an ASCII version of **sysadm** appears as shown in Figure 2–4:

```
                    Main Menu

  1  Session ->          Manage this sysadm session
  2  File System ->      Manage file systems
  3  System ->           Manage DG/UX system databases
  4  Client ->           Manage OS and X terminal clients
  5  Device ->           Manage devices and device queues
  6  Logging ->          Manage system and network logging
  7  Networking ->       Manage the network
  8  User ->             Manage users and groups
  9  Software ->         Manage software packages
 10  Availability ->     Manage high availability features
 11  Help ->             Get help on sysadm and its queries

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

Figure 2–4 ASCII sysadm main menu

Using the graphical interface

To navigate the graphical interface using the mouse, move the pointer to the desired menu and select the menu by clicking the mouse's select button (typically the left button). Continue this way, selecting the desired menus until you reach an operation you want to perform. With the mouse, select the operation to start it.

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

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

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

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

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

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

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

Using the ASCII terminal interface

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

Table 2–3 presents a summary of the methods for making ASCII **sysadm** menu choices.

Table 2–3 Making ASCII sysadm menu choices

User Input	Description
number	Choose menu item by entering *number.*
name	Choose menu item by entering full name of menu item, such as **Session**, or a string fragment that uniquely identifies the menu item such as **Ses** or **ses** for **Session.** The string is not case-sensitive.
names-separated-by - colons	Specify menu traversals by using multiple menu names, with the names separated by colons. Again, the case of characters is not significant. For example, **Software:Package:Install** or **So:Pack:In**.
?	Print help message, then redisplay menu prompt.
*number***?**	Print help message for a particular menu item, then redisplay menu prompt.
q	Exit from **sysadm** from any menu.
New Line/Enter key	Redisplay menu.
^ or ..	Return to the next higher menu.

When an operation presents a query, a default response often appears within brackets. For example, [yes] indicates an affirmative response if you press Enter.

IMPORTANT: When only a predetermined set of responses is appropriate, use the **?** key to display all your choices. You may then select your choice by number or by a unique string.

After you select an operation and enter any information that it requires, **sysadm** either performs the action immediately or asks for confirmation before performing a potentially destructive action. Examples of destructive actions include deleting an OS client or a release area.

Sysadm provides on-line help in several ways. At the main menu, the Help menu offers information on: **sysadm**, the interface, the **sysadm** version, and help itself.

In addition to the Main Menu help option, each menu and operation in **sysadm** has a help message. Enter **?** to get help about the current menu, or enter a menu selection number followed by the **?** key to get information about a particular selection. You may also use **?** to get help and syntax information from any query.

Selecting menu options

In the graphical interface, you select with the:

Mouse	With the mouse pointer on your choice, click button 1 (the left button). Optionally, you may drag the mouse across menus: press and hold the select button while moving the mouse across the desired selections. Releasing the select button when an operation is highlighted begins the operation.
Keyboard	From the Main window, type **alt**-*letter*, where *letter* is the underlined letter (**alt-u** for User, for example). From a pull-down window, type *letter* (**a** to select User Account on the User pull-down menu, for example).
Arrow keys	From the User pull-down, with a box around User Account, press the right arrow key to see the User Account functions.

In the ASCII interface, you can select menu options by entering the number of a menu item or by typing an abbreviation of the menu item name. For example, from the Main Menu, typing "`se`" and pressing Enter selects Session, entering "`sy`" selects System and, entering "`so`" selects Software.

You can start the ASCII interface at any menu level you want to use. For example, to start **asysadm**, skipping directly to the `User-> Group` menu, enter:

```
# asysadm -m User:Group ↵
```

In the discussion of tasks in this manual, to illustrate the selection of the List option on the Group menu, we write: select `User-> Group-> List`. Depending on the interface you're using and your starting point, you interpret these instructions in one of the following ways:

- click User in the **sysadm** main menu, Group in the User pull-down menu, and List in the Group pull-down menu
- enter `8` from the **asysadm** main menu, `2` from the User menu, and `4` from the Group menu
- start the utility with the command:

```
# asysadm -m User:Group:List ↵
```

Controlling the sysadm session

The options in the **sysadm** Session menu let you control message verbosity level (the types and volume of messages recorded) and add comments to the **sysadm** log file, **/var/adm/log/sysadm.log**. Options in this menu also let you exit and restart **sysadm**. For more information, see the contextual helps for the Session menu options.

Getting general and context-sensitive help

For descriptions of general topics, select the **sysadm** Help option. For example, for a description of menu option selection methods and tear-off windows, select `Help-> On Interface`.

For help messages describing particular details, use the **sysadm** context-sensitive help facility. At any **sysadm** prompt, you can either enter the requested information or request an explanation. In the graphical interface, press function key 1 (F1) to request a help message. In the ASCII interface, type a question mark (`?`) and press Enter. (In the ASCII interface, obtain a description of a menu by entering *n*`?`, where *n* is the number of the menu.)

For example, in the process of adding a user, you see the prompt:

```
Login Name:
```

You can enter a value or you can first request an explanation of the prompt by pressing F1 if you are using the graphical interface or by entering `?` if you are using the ASCII interface.

Keeping a window in place to repeat operations

When repeating the same **sysadm** operation from the graphical interface (for example, adding or changing several dozen accounts), you can save mouse clicks by using a *tear-off menu* rather than the equivalent pull-down menu. There is a dashed line at the top of the User pull-down menu. This indicates that the pull-down menu is optionally a tear-off window. If you select the dashed line, the pull-down menu becomes a tear-off menu at the new location and remains in place on your screen until you close it.

With both a pull-down menu and a tear-off menu, you return to the **sysadm** main menu each time you complete an operation.

Using the shell command line to bypass menus

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

sysadm -m *menu-name(s)*

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

You specify menu names using colon-separated lists of strings that are not case-sensitive (see Table 2–3 for definitions). For example:

```
# sysadm -m file:local ↵
```

```
# sysadm -m f:l ↵
```

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

Returning to the shell during a sysadm session

During a **sysadm** session, you can easily return to the shell (Bourne, C, or Korn). How you return depends on whether you are using the graphical or the ASCII terminal **sysadm**.

From graphical **sysadm**, the shell runs as a background job and occupies a separate window. To return to the shell, simply move the mouse back to the shell prompt in the shell window. To return to **sysadm**, move the cursor back to the **sysadm** window.

From an ASCII-based **sysadm** prompt, escape to the shell by executing the appropriate shell escape command. An example of a C shell escape follows:

```
Enter a number, a name, ? or <number>? for help, or
   q to quit: !csh↵
```

Use **!sh** for a Bourne shell escape and **!ksh** for a Korn shell escape.

Return to ASCII **sysadm** by typing:

```
% exit ↵
```

Interpreting sysadm references in this book

The explanations of tasks in this manual use a general format for paths you follow through the **sysadm** menus, regardless of the

interface you use. For example, to build an automatically configured kernel, start at the **sysadm** Main Menu and follow this path:

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

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

Figure 2–5 shows the Kernel Menu in the graphical interface.

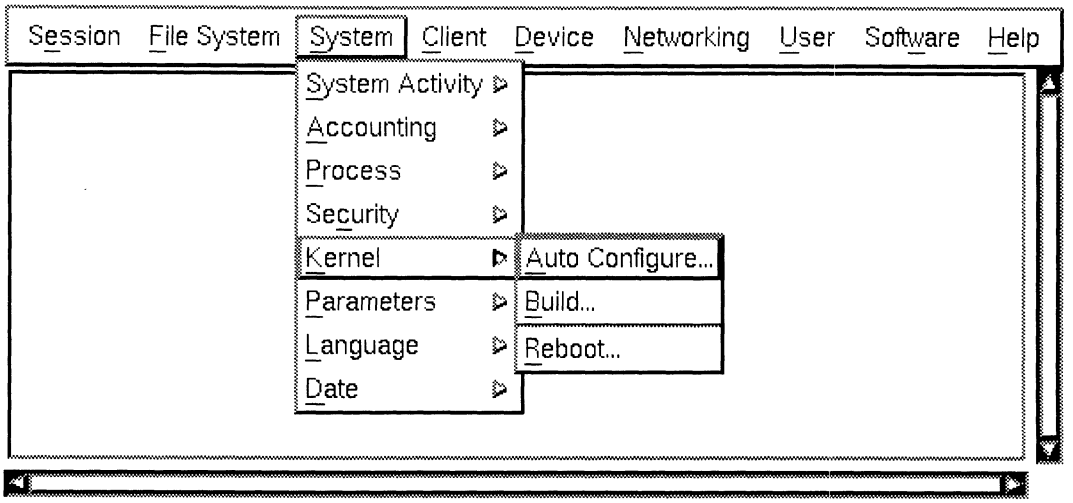


Figure 2–5 Graphical sysadm Kernel Menu

Figure 2–6 shows the Kernel Menu as it appears in the ASCII terminal menu interface.

```
                Kernel Menu

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

  3  Reboot ...            Reboot kernel

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

Figure 2–6 ASCII sysadm Kernel Menu

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

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

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

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

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

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

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

Relationship between sysadm and commands

Data General provides two levels of system management tools in addition to the standard UNIX commands: administrative commands (commands with names beginning with **adm**) and the **sysadm** tool. These system management tools are intended to save you time and help guide you through complex tasks or tasks you perform infrequently. Also, since the tools update system files and perform some input checking before updating files, they help ensure that all necessary files are updated for a given operation and that incorrect content is not introduced into critical files.

There is a strong relationship between **sysadm** and the commands provided with the DG/UX system, as illustrated in Figure 2–7.

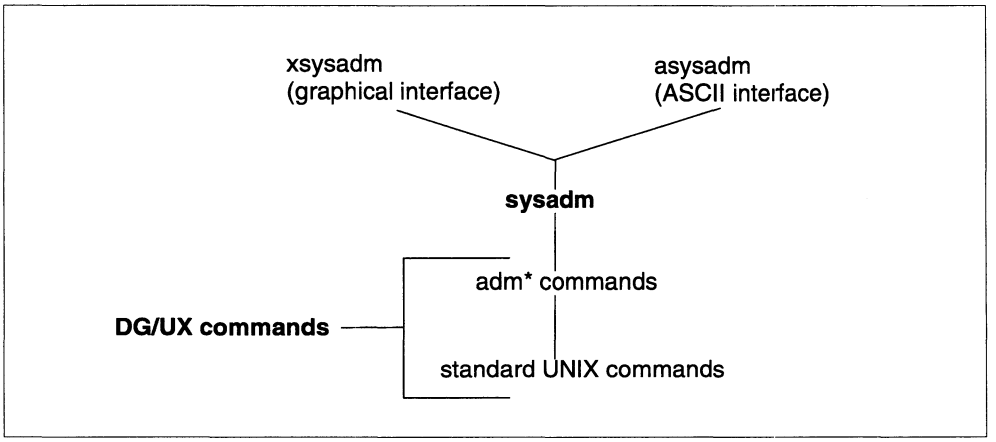


Figure 2–7 Relationship between sysadm and commands

The DG/UX implementation of standard UNIX commands underlies the entire management tools structure. The administrative (**adm**) commands designed and implemented by Data General are most often composed of one or more standard commands organized into a form similar to a shell script. The **sysadm** interface, functionally the same in graphical (**xsysadm**) and ASCII (**asysadm**) forms, provides a menu-driven mechanism that gathers information and invokes execution of the administration commands.

So you have a wide range of choices when you want to perform a system management job on the DG/UX system. You can use standard UNIX commands or DG/UX administrative commands from the command line or in shell scripts. If you prefer to work with a menu-driven interface, you can use **sysadm** from a graphical or ASCII terminal.

You can set up **sysadm** to see the administrative commands invoked by **sysadm** operations by selecting the Session–> Parameters–> Set menu and setting the `Verbosity` option to `Debugging information (actions and commands)`. To view the **/var/adm/log/sysadm.log** file of executed commands and messages maintained by **sysadm**, you can use the **sysadm** Session–> Log File–> List Log File operation. You can copy the contents of this log file to use as a basis for shell scripts to automate your work.

To generate a file containing a list of the administration commands provided by Data General, use the following command:

```
apropos adm > admcommands
```

The **admcommands** file will contain a list of DG/UX commands containing the string "adm" in their names or descriptions. The commands with names beginning with "adm" are the administrative commands.

Stand-alone sysadm

A stand-alone **sysadm** utility is also available for use with the DG/UX system. The stand-alone **sysadm** pathname is **/usr/stand/sysadm**. You can issue shell commands directly to the stand-alone **sysadm** for tasks such as disk repair.

For how to use stand-alone **sysadm**, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

IMPORTANT: Data General no longer ships the **diskman** stand-alone management tool with DG/UX software. The two versions of **sysadm** include all functions of the **diskman** utility.

Typical AViiON computer configurations

In this book, you will come across terms that refer to the roles played by AViiON® computers running the DG/UX system. In general, a computer that provides network service (such as the operating system or file service) is a server. Correspondingly, computers that receive such services are considered clients.

In the DG/UX environment, these are the most common configurations of AViiON computers:

- OS server

  This is an AViiON computer that provides user services through direct terminal lines and/or a local area network (LAN). The server provides operating system (OS) resources through direct terminal access or to client computers across a LAN. In addition to offering OS resources, a server might also provide file service.

  You can manage the DG/UX system through either a system console (an asynchronous terminal with a keyboard) or a workstation's system console, which consists of the graphics monitor and keyboard. A workstation is an AViiON computer that provides graphical capabilities.

- OS client

  This is an AViiON computer that may or may not have its own disk; thereby relying on the OS server for some or all of its operating system software and file service. Once the OS server establishes a connection with the OS client (by transferring a kernel image), the OS client subsequently can be responsible for booting its OS over the network and mounting remote file systems.

  If an OS client does have an attached disk, it can accommodate some or all of its own operating system. There are various hybrid OS client configurations available for improving system performance. For information on these configurations, see Chapter 18.

  After OS services have been provided to the OS client, the user of the OS client machine assumes certain system administrative responsibilities such as maintenance of the local disk, mounting remote file systems, booting the system, and taking the system down.

- Stand-alone workstation

  A workstation is an AViiON computer that provides graphical capabilities. When it stands alone, it has an attached disk subsystem that provides its own OS resources. However, a network connection is often desired for access to remote file systems and communications services such as electronic mail.

End of Chapter

3 Booting, rebooting, and shutting down the system

This chapter explains booting and rebooting the system, shutting it down, and handling boot time problems. The chapter also explains what happens while the system boots and the system components that control booting.

The primary reasons for booting or rebooting a DG/UX system are to restart the system following a normal or abnormal shutdown and to gain access to a new kernel that has just been built.

Booting and rebooting your computer kills all running processes. If an OS server reboots, attached OS clients may continue to operate normally or they may reboot automatically. Rebooting an OS client does not affect the OS server.

Warning users before rebooting or shutting down

If users are logged in to the system before you reboot or shutdown, broadcast a message warning that you are about to reboot or shutdown the system. See the **wall** manual page for information on sending a broadcast message to all users logged in the system. Make sure all users are logged out before proceeding. An example of such a warning follows.

```
# /etc/wall ↵
Five minutes until we reboot the system. Please log off. ↵
<Ctrl-D>
```

Rebooting from sysadm

To reboot a system, follow this path through **sysadm**:

```
System-> Kernel-> Reboot
```

The system prompts you for a boot path:

```
Boot path: [sd(insc(0),0,0)root:/dgux -3] ↵
```

The default boot path is used to boot the system unless you specify an alternative. The boot path identifies the boot file located on a local device (disk or tape) and, optionally, the run level to which the

system will boot. A default boot path is preset at the factory. You can reset the default with the **dg_sysctl** command or the SCM Configuration Menu. If the boot path does not include a run level, booting brings the system up to the default run level set in **/etc/inittab**.

If you accept the default boot path in the preceding example, you will boot the kernel **/dgux** (the boot file) on the **root** virtual disk located on the device **sd(insc(0),0,0)** to a run level of 3.

You are then asked for confirmation:

```
All currently running processes will be killed.
Are you sure you want to reboot the system? [yes] ↵
```

The following type of message indicates that the boot process has begun:

```
Booting sd(insc(0),0,0) loading...
```

The login prompt appears after the DG/UX system has finished booting. At this point, you and your users can login the system with any defined user login.

Booting from the SCM prompt

A system that has been shut down operates at the SCM level. To start the DG/UX system at the SCM prompt, use the **b** (for **boot**) command. An OS client will boot its kernel from the SCM.

From the `SCM>` prompt, issue a boot command using this syntax:

`SCM>` **b** *boot-path*

where **b** stands for boot and *boot-path* identifies the boot file located on a local device (disk or tape) and, optionally, the run level to which the system will boot.

IMPORTANT: AViiON systems can boot only from media formatted with a fixed-length record size of 512 bytes. Therefore, AViiON systems will not boot from QIC-320 or QIC-525 media written to in either 1024 bytes (1Kbyte) block size or in variable block size mode.

The *boot-path* is made of the following components:

DG/UX-device-name virtual-disk:*pathname* [*–run-level*]

where:

DG/UX-device-name	identifies the boot device. You can specify devices using the short or long form.
virtual-disk	contains the executable kernel image.
pathname	is the pathname of the executable kernel image on the virtual disk.
-run-level	specifies the run level to which your system boots automatically. You should set your run level in the **/etc/inittab** file and use the boot path to override the pre-established default. For information on run levels, see "DG/UX run levels." For information on editing the **inittab** file, see "The /etc/inittab file."

IMPORTANT: Use spaces only between *pathname* and *–run-level*.

The boot path in the following example boots the **/dgux** kernel contained on the virtual disk **root**. The boot device is a SCSI disk attached to the first integrated SCSI adapter whose SCSI ID is 0. The system boots to a run level of 3, which enables multiuser mode.

```
SCM> b sd(insc(0),0,0)root:/dgux -3
```

The boot path in the next example boots the first virtual file (0) on a 150-megabyte QIC tape whose SCSI ID is 4:

```
SCM> b st(insc(0),4,0) ↵
```

The following boot command automatically boots the file identified in the autoboot path that you set through the SCM Configuration Menus:

```
SCM> b ↵
```

You can also enter the preceding boot command (with no *boot-path*) if you set the default boot path with the SCM's **f** (for **format**) command:

As another example, to boot from the first SCSI disk on an AV5220 computer system, use this command line:

```
SCM> b sd(cisc(),0)root:/dgux ↵
```

CAUTION: If your system is part of a dual-initiator configuration and shares a SCSI bus with another system, be careful to specify the correct SCSI adapter SCSI ID in the boot path. If the boot path you use to boot your system specifies the SCSI ID of the SCSI adapter installed in the other system, the boot will fail and the SCSI bus will hang. If the SCSI bus hangs, an attempt by either system to access the SCSI bus will hang the system. When this happens, recover by resetting the hardware and rebooting. For information on failover disks, see Achieving High Availability on AViiON® Systems.

The login prompt appears after the DG/UX system has finished booting. At this point, you and your users can login the system with any defined user login.

Booting a stand-alone executable file

In the examples of booting from the SCM prompt, we loaded the standard operating system. However, you can use the **boot** command to load any stand-alone, machine-executable file you choose. A stand-alone, machine-executable file is one that can run directly on the AViiON system hardware without an operating system.

As shipped, the DG/UX system's bootable files are:

/dgux
A kernel configured for all devices in the standard locations.

/dgux.installer
A hard link pointing to the same file as **/dgux**.

/dgux.starter
A hard link pointing to the same file as **/dgux** and **/dgux.installer**.

/usr/stand/sysadm
Stand-alone **sysadm**, used for disk management operations when the system on disk is unavailable or for operations on the system disk itself.

If, for instance, you want to boot stand-alone **sysadm**, **/usr/stand/sysadm**, located on the first SCSI disk, you would execute it with a command line such as:

```
SCM> boot sd(cisc(),0)usr:/stand/sysadm ↵
```

You can boot files that reside on file systems built on multiple partitions (to form an aggregation) as long as they are located on

one physical disk. Such an aggregation cannot span multiple physical disks. For example, if your **/usr** file system were built on a virtual disk comprised of more than one partition, you could boot **/usr/stand/sysadm** as as long as the partitions were located on the same physical disk.

Booting an OS client over a LAN

An OS client boots its kernel via a local area network. The command syntax is:

network-controller-device OS-server Internet-address:pathname [*–run-level*]

IMPORTANT: Do not use spaces within any of the two fields.

where:

network-controller-device
: is the name of the device used to connect the OS client to the local area network. Valid controller names have the form *controller-type* (*controller-number*) where *controller-type* can be **inen** (integrated Ethernet controller), or **dgen** (Data General second generation integrated Ethernet controller). The *controller-number* for controller type **inen** and **dgen** can be only **0**. Valid examples are **inen(0)** or **dgen(0)**.

OS-server-Internet-address
: is the Internet address of the OS server. An example of an Internet address is **128.223.2.1**.

pathname
: identifies the kernel image to boot.

[*–run-level*]
: specifies the run level to which your system boots automatically. You should set your run level in the **/etc/inittab** file and use the boot path to override the pre-established default. For information on run levels, see "DG/UX run levels." For information on editing the **inittab** file, see "The /etc/inittab file."

In the following example, the OS client boots the file located in **/srv/release/PRIMARY/root/bob/dgux** over the network controller **inen**( ) by way of the OS server's Internet address, and **bob** is the OS client hostname.

inen() 128.223.2.1:/srv/release/PRIMARY/root/bob/dgux ↵

If there are OS clients booting for the first time, you need to set up software packages and mount file systems before their computers can do useful work. For information on OS clients, see Chapter 18.

Booting alternate root and swap areas

To boot an alternate virtual disk for **root** and **swap**, include the **–q** option on the boot command line. The command prompts you for the names of the virtual disks to be used in place of **root** and **swap**. For example:

```
SCM> b sd(cisc(),1)root2:/dgux -q ↵

Booting sd(cisc(),0)root:/dgux  -q
Swap disk name (or q to quit)? [swap]  newswap ↵
Root disk name (or q to quit)? [root]  root2 ↵
```

If the kernel cannot find the default virtual disks that you specified (perhaps because of an accidental deletion), the system attempts to find virtual disks named **swap** and **root** on the disk from which you booted. If that fails, the system will attempt to find virtual disks **swap** and **root** on any registered disk. If this fails, the system prompts you again for the names of the **swap** and **root** virtual disks to use.

Alternatively, you may set a default **root** and **swap** virtual disk to be booted from a given physical disk. You may establish default boot settings for use with the **dg_sysctl** command, which initiates an unattended automatic system reboot in the event of a panic.

Build the virtual disks that you want to use as **root** and **swap**; for example, assume you have **root_production**, **root_test**, and **swap**. If you normally prefer booting **root_production** and **swap**, you could set boot defaults on the physical disk using the **admpdisk** command, as follows;

```
#admpdisk -o set_defaults -r root_production -s swap
'sd(ncsc(),0)'↵
```

This enables you to set your SCM boot path without specifying the **root** virtual disk or the **–q** option. An example follows:

```
SCM> b sd(ncsc(),0)/dgux -3 ↵
```

To reboot the system using **root_test**, reset the defaults on the physical disk using the following command:

```
# admpdisk -o set_defaults -r root_test 'sd(ncsc(),0)'↵
```

You may reboot without specifying the **root** virtual disk or the **–q** option.

When you boot the system from a physical disk that contains default settings, the bootstrap will attempt to find the named kernel image (**dgux** if it is not specified on the command line) on the virtual disk that is associated with the default **root** virtual disk.

During kernel initialization, the system looks for default settings on the disk from which the system was booted. If it finds settings, it mounts the default **root** virtual disk as the / file system, and begins swapping on the default **swap** virtual disk.

You can remove the **root** and **swap** default virtual disk settings by specifying a null argument in the form of empty double quotes. For example:

```
# admpdisk -o set_defaults -r " " -s " " 'sd(ncsc(),0)'↵
```

To set different boot defaults on different physical disks, maintaining multiple copies of DG/UX on your system, we recommend that you install the DG/UX system on one physical disk using names such as **root_1**, **swap_1**, **usr_1**, and so on. On another physical disk, for another version of the operating system, adopt another naming convention: **root_2**, **swap_2**, **usr_2**, and so on. We strongly urge you to assign unique names to each set of virtual disks to avoid confusion. For example:

```
# admpdisk -o set_defaults -r root_1 -s swap_1 'sd(ncsc(),0)'↵

# admpdisk -o set_defaults -r root_2 -s swap_2 'sd(ncsc(),1)'↵
```

To boot from the first root, use the command line:

```
SCM> b sd(ncsc(),0)↵
```

To boot from the second root, use the command line:

```
SCM> b sd(ncsc(),1)↵
```

The physical disk has associated with it the virtual disks containing the kernel image, the / file system, and swap space.

IMPORTANT: You cannot assign alternate locations for the **usr** virtual disk since it is mounted after the kernel initializes. If you have multiple **/usr** file systems, you must set up the proper **/etc/fstab** entries in each of the root file systems you will be booting, referencing the corresponding **/usr** file system to use.

Booting from mirrors and caches

Although you can boot only from virtual disk partitions and aggregations, **root** and **usr** can exist as other types of virtual disks as well, such as a mirror or a cache. However, you cannot boot directly from a cached or mirrored **root** or **usr**. Instead, you need to supply one of their children as the virtual disk from which to boot.

If **root** or **usr** is a mirror, you need to boot from one of the images of the mirror as follows:

```
SCM> b sd(insc(),0)root_image1:/dgux
```

If **root** or **usr** is a cache, you need to boot from the back-end cache device as follows:

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

The child from which you boot the mirror or cache must be a bootable virtual disk (a partition or multi-piece aggregation).

For information on mirrors, caches, and virtual disk partitions and aggregations, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Rebooting from the DG/UX command line

You can use the **reboot** command to halt the system and boot it without placing you in the SCM:

```
# reboot ↵
```

The **reboot** command, without arguments, boots the system using the boot command line used the last time the system booted. Optionally, you may specify a different boot path.

Another way to reboot is to use **init** to change run level to **6**, which is the same as executing the **reboot** command. See the **reboot** manual page for more information.

The login prompt appears after the DG/UX system has finished booting. At this point, you and your users can login the system with any defined user login.

Rebooting automatically after a power failure

After a power outage, the automatic boot mechanism reboots your system without operator intervention, bringing it 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:
```

For more information about run levels, see "DG/UX run levels."

Verifying that the system is initialized and functioning correctly

While the system boots, it produces a variety of messages, log files, and other output that you can review to verify that your system is initialized and functioning correctly. The following sections describe briefly how to review the information produced at boot time.

Monitoring the boot process

When you boot the system, messages describe the progress of system initialization. These messages start with the loading of the kernel and end when the system has reached the run level to which you have set your system to boot. It is good practice to watch these messages as booting occurs, in case an error message appears. As an alternative, you can review the boot messages later by looking at the file **/etc/log/init.log**.

The first messages result from the loading and initialization of the kernel and reflect the current installed version of the DG/UX system. The first messages also tell something about your hardware, the amount of physical memory configured in the system and the number of processors making up the CPU.

The kernel then configures hardware devices on your system. The kernel recognizes only those devices for which you have included a device driver specification in the system file when you built the kernel. For more information on building kernels, see Chapter 15.

After the kernel has initialized itself, it starts the **init** program to continue system initialization and to start system services. The **init** program performs these functions by executing a series of check scripts and setup scripts, all of which produce their own output. The output messages on a typical system with networking might look like this:

```
Checking local file systems ...
Current date and time is Tue Sep 22 14:30:24 EDT 1992
Checking system files .....
Enabling automatically pushed STREAMS modules .....
Linking short names for /dev device nodes ...
Loading terminal controllers ....
Starting disk daemons .....
Mounting local file systems ..........
Checking for packages that have not been set up ...
Starting miscellaneous daemons ...
Starting Logical Link Control Services ...
Starting TCP/IP network interfaces .........
Starting system logging daemon ....
Starting NIS services ......
Starting NFS lock services ......

NOTE: Pausing for 15 seconds to allow remote systems to
    reclaim NFS locks.

Starting batch services ....
Starting line printer scheduler ....
Saving ex(1) and vi(1) temporary files ....
Starting NFS services .....
Starting TCP/IP daemons ........
Mounting NFS file systems .....

NOTE: See /etc/log/init.log for a verbose description of
     the system initialization process.
```

In general, it is good practice to look for error messages that may appear in the boot display. In addition, there are several areas where you should pay particular attention.

Checking local file systems

The file system checker, **fsck**, runs every time you boot the system. When **fsck** runs at boot or during a change of run levels, its output goes to **/etc/log/fsck.log** or **/etc/log/fast_fsck.log**.

The system checks file systems only if an unexpected event such as a system crash or a disk or disk controller hardware failure causes service to the file system to terminate abnormally. In these cases, the system will not allow further access to the file system until you have checked it with **fsck**. If you do not wish to reboot the system to start file system checking, you can use the **sysadm** utility or **sysadm**'s File System menu to start **fsck**. For more information about **fsck**, see the **fsck** manual page and *Managing Mass Storage Devices and DG/UX™ File Systems*. For information about file

checking and **sysadm**'s File System menu, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Checking system files

The **chk.system** script performs several functions, including checking for package setup scripts that have not executed, verifying that the DG/UX parameters file is available, verifying that the **/tmp** directory exists and is usable, and adding additional swap areas if defined.

One important function of this part of the boot process is to check for user profiles that do not have passwords. A user profile without a password allows anyone to log in using that username. If such profiles exist, you should assign passwords to them immediately. The **sysadm** User menu provides operations for managing user profiles.

Initially, the DG/UX system has no passwords for the superuser profiles **root** and **sysadm**. It is important for you to assign passwords to these profiles as soon as you install your system. Leaving these profiles without passwords allows anyone to log in with superuser access.

If you installed the X Window System package, there will also be an **xdm** profile that does not have a password. Although the system warns you of this condition at every boot, you do not need to provide a password for **xdm** (it does not constitute a security hazard).

User profiles are in the **passwd** file, located in **/etc**. The **passwd** file is readable by all. Although the passwords are encrypted, any user on your system can identify profiles that have no password.

Checking for packages

This part of the boot process checks for software packages that are not set up. This check only includes software packages loaded using the utilities in **sysadm**'s Software menu. Typically, these packages require that you load them onto disk and then set them up before you can use them. There is nothing wrong with having a package that is not yet set up; this check simply serves as a reminder for you. You cannot use a package until you set it up. To set up a package, use the utilities in the Software menu.

If this check finds that you have loaded but not set up a release of the DG/UX system software, it begins the setup process without prompting.

Checking /etc/log

A number of the services started at boot leave logs in the **/etc/log** directory. After every boot, it is good practice to check this directory and review the logs.

The **ls** command provides options that allow you to look more easily for log files created during the last boot. After changing to the **/etc/log** directory, execute this command to list files in the order they were last changed:

```
# ls -ltr ↵
```

The most recently changed files appear at the bottom of the listing. If you check this directory after every boot, you soon learn to tell if a file contains unusual information simply by looking at the size of the file.

As the system creates boot logs, it renames boot logs left from the previous boot. It renames them by adding the suffix **.old** to the file name. In the process of renaming the previous boot logs, any existing logs with the **.old** suffix are deleted.

Some log files you may want to review are:

init.log
This file contains messages that appeared during the boot process. The **chk** and **rc** scripts found in **/usr/sbin/init.d** produce these messages.

fsck.log and **fast_fsck.log**
When run at boot or during a run level change, **fsck** produces these log files. When **fsck** finds that a file system is consistent, it logs an entry like this:

```
/dev/rdsk/disk: No check necessary for /dev/rdsk/disk.
```

where *disk* is the name of the virtual disk containing the file system.

When file systems are corrupt, the **fsck** log contains messages for each error found in the file system. For a discussion of **fsck** and its output, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

nfsfs.log
This file contains the output from the **mount** command, which indicate whether or not the attempts to mount directories failed or succeeded. When the mount fails, the entry includes the error message. To diagnose network-related failures, see *Managing TCP/IP on the DG/UX™ System* or *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

Checking lost+found

One of the functions of the **fsck** file checker is to locate blocks of data that have become disconnected from their files. If the **fsck**

utility cannot reconnect the data to its file, it puts the data in its own file and puts the file in a directory called **lost+found** in the file system's mount point directory. The mount point directory is the directory where you have attached the file system to your system's directory structure.

For example, if you have a file system mounted on mount point directory **/sales/accounts**, **fsck** puts any disconnected blocks in files in **/sales/accounts/lost+found**. The files in this directory have names reflecting where **fsck** found them.

It is good practice to check the **lost+found** directory of a file system after **fsck** has checked it. You could use a simple script like the following to list files in all mounted local **lost+found** directories:

```
#!/bin/sh
/etc/mount | /bin/grep ' type dg/ux ' | /bin/cut -d" " - |\
while read DIR
do
  if test -f $DIR/lost+found/*
  then
    echo Found lost file fragments in $DIR/lost+found:
    /bin/ls -l $DIR/lost+found/*
  fi
done
```

You may also find the **file** command helpful. This command determines the nature of a file by classifying it as English text, data, binary executable code, and so on.

Example: a power outage causes your system to crash. When you reboot the system, **fsck** begins checking file systems. When **fsck** is finished, you look in **fsck.log** (or **fsck_fast.log** for file systems mounted for fast **fsck** checking) and see that file system **/sales/accounts** required checking by **fsck**. The file system is now mounted and accessible. You do this:

```
# cd /sales/accounts/lost+found ↵
# ls -l ↵
total 8
-rw-rw-rw-  1 curly  sales    3273 Sep 23 1991 #177431
# file #177431 ↵
#177431:  English text
```

You see that **fsck** found a piece of a file belonging to user **curly**. The **file** command classifies the contents as English text. Now you can tell user **curly** that one of his files was damaged, and, using the fragment from the **lost+found** directory as a clue, he can set about determining what file was damaged so that he can repair it or have it restored from backup.

Depending on how you configure your system, rebooting after a power outage or other failure may occur without operator intervention. Thus, file system checking could occur without your ever realizing it. Therefore it is good practice to check the **lost+found** directories periodically to be sure there are no file fragments there. The absence of a **lost+found** directory simply means that **fsck** has never needed to create one.

Shutting down the DG/UX system

Before shutting down the system, broadcast a message warning users that you are about shut down. See the **wall** manual page for information on sending a broadcast message to all users logged in the system. An example of such a warning follows:

```
# /etc/wall ↵
Five minutes until system shutdown. Please log off. ↵
<Ctrl-D>
```

To shut down the system, become superuser and enter these commands from the system console:

```
# cd / ↵
# shutdown -g30 -y ↵
```

The first command changes the current directory to / (root). The second command specifies a grace period of 30 seconds before shutdown begins. Also, it includes an affirmative response (yes) to start the shutdown. Otherwise, a confirmation request appears, requiring a response.

The following messages appear when the shutdown process starts and finishes:

```
Shutdown started.    Wed June 16 11:08:57 DST 1993
...
Shutdown is complete.
```

Once the system is shut down, enter:

```
# halt ↵
```

The **halt** command transfers system control from DG/UX to the SCM. When the SCM prompt appears, the DG/UX system has shut down.

Shutting down the system means taking it to a lower run level. Often, you take the system to run level 1, the administrative state,

to perform certain administrative tasks. At other times, you may want to shut down the system so you can halt the processor. In either case, you use the **shutdown** command. In single-user mode, you can use the **halt** command to stop the processor.

When you shut down the system, system buffers are flushed, open files are closed, user processes and daemons are stopped, file systems are unmounted, and file system superblocks are updated. For what happens as the system comes down through run levels, see "DG/UX run levels."

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

Shutting down to a lower run level

Let's assume we're currently in run level 3 (multiuser mode) and we want to go down to run level 1 (administrative mode). 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 that shutdown will continue without further 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 shutdown the system to run level 1:

```
# 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

You can shut down to run level S (single-user mode) from any other level. This example shows a shut down from run level 1. Type:

```
# cd  / ↵
# shutdown  -g0 -y ↵
```

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

```
# halt  -q ↵

CPU HALTED


SCM>
```

You can also halt the system by using **init** to change to run level **0** or **5**. Once at the SCM prompt, you may turn off power to the computer.

Software power shut-off after shutdown

Some Data General AViiON computers support a feature allowing you to shut off power to the system using a software command. Using this feature, you can set the system to shut off power automatically after a normal shutdown. You cannot set the system to shut off power after a panic.

On systems that support this feature, use the **-p** option of the **dg_sysctl** command to set up the system to shut off power after normal shutdown. For example:

```
# dg_sysctl -p auto ↵
```

The **dg_sysctl** command ignores the **-p** option if your system is set for automatic boot (as with **-r auto**). On systems that support the power-off feature, the default is **auto**; on systems that do not support this feature, the default is **skip**.

Setting the default boot path

You can establish a shortcut for booting your DG/UX system by setting a default boot path for the kernel. The shortcut allows you to type only the **b** command in lieu of a longhand boot command from the SCM prompt.

For an explanation of boot path syntax, see "Booting from the SCM prompt."

1. Type the **f** (format) command at the SCM prompt. The SCM Configuration Menu appears:

```
SCM> f ↵

View or Change System Configuration

1. Change boot parameters
2. Change console parameters
3. Change mouse parameters
4. Change printer parameters
5. View memory configuration
6. Change testing parameters
7. Return to previous screen

Enter choice(s) ->
```

2. Type the option **1** followed by Enter to change boot parameters. The following menu appears.

```
Change Boot Parameters

1 Change system boot path
2 Change diagnostics boot path
3 Change data transfer mode [BLOCK]
4 Return to previous screen

Enter choice(s) ->
```

3. Type the option **1** again, followed by Enter. The system displays the current boot path. Enter **y** followed by Enter to change the path.

```
System boot path = [sd(insc(0),0,0)root:/dgux.installer]

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

4. Enter the new boot path at the prompt. The system them prompts for confirmation.

Two boot disk examples follow. The first one is for a system having a SCSI system (boot) disk; it boots to a run level of 3 automatically.

```
Enter new system boot path -> sd(insc(0),0,0)root:/dgux -3 ↵
System boot path = [sd(insc(0),0,0)root:/dgux -3]
Do you want to modify the system boot path? [no] ↵
```

The next example is for a system having an ESDI system disk; it boots to a run level of 3 automatically.

```
Enter new system boot path -> cied( )root:/dgux -3 ↵
System boot path = [cied( )root:/dgux]
Do you want to modify the system boot path? [no] ↵
```

The system then offers to boot via the path you specified:

```
Do you want to boot [N]
```

5. If you want to reboot immediately, answer **yes** to the next question. Otherwise, accept the **N** default, and return to the SCM prompt.
6. The Change Boot Parameters menu appears. Choose option 4.

   The View or Change System Configuration menu is appears. Choose option 7 to go to the SCM prompt.

   From now on you can type the letter **b** followed by Enter at the SCM prompt to boot the kernel.

Setting the automatic reboot behavior to handle panics

You can use the **dg_sysctl** shell command to define your system's booting behavior following a panic situation. You can establish whether or not the DG/UX system will automatically reboot and identify the mass-storage device to receive a system dump.

The syntax follows:

dg_sysctl [**–t**][**–r** *reboot-state*][**–b** *"boot-path"*][**–d** *autodump-state*] [**–f** *"dump-device"*]

where:

–t Makes a temporary change. The change does not persist following the system reboot. The default specifies a permanent change.

–r Sets the system's reboot behavior. If *reboot-state* is **auto**, the system automatically reboots after a panic. If *reboot-state* is **halt**, the system does not automatically reboot after a panic. The default is **halt**.

–b Sets the system's boot path. The boot path must conform to the SCM boot syntax. The *boot-path* is the path to use when the system is rebooted. Be sure to surround the boot path with double quotation marks (" "). For an explanation of boot path syntax, see "Booting from the SCM prompt."

–d Sets the system's autodump behavior. If *autodump-state* is **auto**, the system attempts to dump to *dump-device* after a panic. A tape must be present in the drive in the event of a dump. If *autodump-state* is **skip**, the system does not attempt to dump to *dump-device* after a panic. If *autodump-state* is **ask**, the system asks if you wish to take a system dump after a panic. The default is **ask**.

–f Sets the system's *dump-device* to be used during a panic. The default value for the DUMP variable is set in the **/usr/etc/master.d/dgux** file and can be reset in the system configuration file. For an AViiON 4000 OS server, for example, the default boot tape device is **st(insc(0),4,0)**. For an OS client, a dump is submitted over the network **inen()** to the OS server's dump device.

dump-device
is the name of the DG/UX device to which a panic dump is written.

Viewing the currently-defined automatic reboot behavior

Entered with no arguments, the **dg_sysctl** command reports the current values for the arguments that it can set:

```
# dg_sysctl
```

The values determine the boot path, whether or not the boot path is permanently changed, if the system will automatically autoboot following a panic or halt and query you about taking a dump, and the dump device.

Enabling automatic reboot and dumping after a panic

The following example of the **dg_sysctl** command enables auto-rebooting after a panic:

```
# dg_sysctl –r auto –b "sd(insc(0),0,0)root:/dgux –3" –d auto –f "st(insc(0),4)"
```

This example reboots the kernel located at the specified boot path to a run level of 3. The command also enables auto-dumping after a panic. The dump goes to **st(insc(0),4)**, a SCSI tape device with a

SCSI ID of 4 that is attached to the first (0) integrated SCSI controller.

Make sure you have an appropriate tape inserted in the drive of the dump device. If the system panics, a dump is written to the specified device automatically.

Setting DG/UX parameters that control the boot process

Various DG/UX parameters control what actions occur on your system when it boots. These actions include checking file integrity, cleaning up after unfinished jobs of various kinds, and starting programs that provide various system services.

To display these parameters, which appear in **/etc/dgux.params**, use the **sysadm** operation `System-> Parameters-> Get`. To turn boot actions on or off, use the operation `System-> Parameters-> Set`.

A list of the actions available through the Set operation appears below, followed by a list of DG/UX parameters not accessible through the Set operation. There is a help message for each parameter in the Set operation.

`Print Verbose Messages at Boot`
: Select this feature if you want verbose messages during each run level change. The verbose messages are always written to the **/etc/log/init.log** file. If you do not want to see the verbose messages, leave this feature set to its default value, off.

`Check Password File at Boot`
: Select this feature if you want the system to check the password file, **/etc/passwd**, for entries that lack passwords each time the system boots. Leave this feature unselected if you do not want the system to look for profiles without passwords.

`Check UUCP Files at Boot`
: Select this feature if you want the system to verify that UUCP file permissions are correct each time the system boots. The system corrects any inappropriate permissions. If you do not use UUCP or if you are confident that the file permissions are correct, do not select this feature.

`Check for Packages Needing Setup at Boot`
: Select this feature if you want the system to check at boot time for software packages that you need to set up. You may want to use this feature if you frequently add new software

packages. If you seldom add new software packages, you may not want to select this feature. If you do not select this feature, the system will not alert you that you have to set up new packages.

`Start File System Checker Without Verification at Boot`
Select this feature if you want the system to start checking the integrity of file systems without querying you each time the system boots. If you do not select this feature, the system will ask you every time it boots if you want to check file systems. If you do not select this feature, realize that the boot process will not continue until you have entered a response at the system console. The system uses **fsck**(1M) to check the integrity of file systems.

`Boot without Verifying Date and Time`
Select this feature if you want the system to boot without asking you to verify the date and time. Leave this feature unselected if you want the system to pause during boot and ask you to verify the system date and time. If you do not select this feature, realize that the boot process will not continue until you have typed some response at the system console. You may use the **date** command to set the date and time while the system is running.

IMPORTANT: Setting the date and time while the system is at run level 1 or higher may cause some system services (or daemons) to behave improperly. For best results, take your system to single-user mode before setting date and time.

`Download Async Ports at Boot`
Select this feature to load and start asynchronous controllers at boot. If you do not select this feature, your system terminals (and any other asynchronous devices, for example, some printers) will not be available when the system comes up. The system uses the **tcload**(1M) command to load asynchronous controllers.

`Mount Local File Systems at Boot`
Select this feature to mount all local file systems at boot. Leave this feature unselected if you have no local file systems or if you do not want them mounted at boot. Local file systems are the ones listed in **/etc/fstab** that are not of type **nfs**. Mounting a file system makes it available for use on the system.

`Start wmtd Daemon at Boot`
Select this feature to start the **wmtd**(1M) daemon at boot. The **wmtd** daemon allows you to use a WORM (write once,

read many) device as a tape device. If you have a WORM device, you may want to select this feature. If you want to specify any arguments for the **wmtd** command line, specify them at the `Arguments to wmtd` prompt in this operation. Leave this feature unselected if you do not have a WORM device.

`Start System Error Log Daemon at Boot`
Select this feature to start the system error log daemon, **syslogd**, at boot. We recommend that you start **syslogd** at boot because some system facilities (such as the disk mirroring portion of the kernel) use it. For information on the system error logger, see Chapter 7.

`Start System Accounting at Boot`
Select this feature to start system accounting each time the system boots. Make sure the appropriate **cron** job entries appear in the superuser's **crontab** file. For information on the accounting system, see Chapter 16. For information about **cron**, see Chapter 6 and the **cron** manual page.

`Start cron Daemon at Boot`
Select this feature to start the **cron** daemon each time the system is booted. You should run this daemon unless you are sure that none of your system's users or packages are using this service. This service is useful for running commands submitted either via a **crontab** file or with the **at** or **batch** commands. If you use the accounting system, the LP (printer) subsystem, or the UUCP facility, you must also run the **cron** daemon. For information about **cron**, see Chapter 6 and the **cron** manual page.

`Start lpsched Printer Scheduler at Boot`
Select this feature to start the **lpsched** printer scheduler daemon each time the system is booted. This daemon must be running in order for users to print jobs with the **lp** command. This daemon also supports print requests submitted with the **lpr** command. If not started at boot time, you can start this daemon later with **sysadm**.

`Start lpd Printer Scheduler at Boot`
Select this feature to start the **lpd** printer scheduler daemon each time the system is booted. This daemon supports print requests submitted with the **lpr** command.

`Preserve Editor Temporary Files at Boot`
Select this feature to run the **expreserve** daemon each time the system is booted. **expreserve** saves the temporary files that are left behind when a system crash or other interruption causes **ex** or **vi** editing sessions to quit prematurely. If necessary, the **expreserve** daemon sends

mail to users telling them how to restore interrupted editing sessions. If any users on your system use **ex** or **vi**, you should run this daemon.

`Number of biod daemons`

Enter the number of **biod** daemons to be started when the system boots. **biod** daemons are used in performing asynchronous I/O between secondary storage and main memory except for paging to local swap areas. For example, file read-ahead, most file buffer write-backs, and paging to a remote disk all use **biod** daemons. The more I/O expected on the system, particularly ONC/NFS I/O, the more daemons are needed to service it. A good value for a typical system using ONC/NFS is 8; fewer **biod** daemons may yield equally good performance if the system is not used as an ONC/NFS client (that is, if it does not access remote file systems much).

`Arguments to swapon`

Enter arguments to the **swapon** command. The **swapon** command runs each time you boot the system, checking the **/etc/fstab** file for entries of type **swap** and making those virtual disks available to the system as additional paging area. You may specify the **-a** option to specify that the system should use all swap areas appearing in **/etc/fstab**, or you may specify particular virtual disks (for example, **/dev/dsk/swap_alt**). Normally, the argument is a null string, which specifies the one default swap device.

`Arguments to wmtd`

Enter the virtual-to-physical device mapping used by the WORM-as-magnetic-tape server daemon. For example, if you want **/dev/wmt/0** and **/dev/wmt/0n** to be associated with the device **/dev/rpdsk/2**, then enter **0=/dev/rpdsk/2**. See **wmtd** for more information. Refer to the file **/etc/devlinktab** to see how virtual devices map to physical devices on your system. If you do not have a WORM device on your system, you may ignore this query.

`Arguments to expreserve`

Enter the names of the directories where the **vi** and **ex** editors create temporary files during editing sessions. The **expreserve** daemon checks these directories for files left by prematurely terminated **ex** and **vi** sessions. **expreserve** moves these abandoned temporary files to **/var/preserve**.

Some DG/UX parameters are not accessible through the Set operation. To change these, edit the **/etc/dgux.params** file. The file contains comments to help you understand the parameters and the accepted values.

The parameters relevant to booting that you change by editing **/etc/dgux.params** are:

dkctl_START
: This parameter determines whether or not the system will enable the write-verify mode of operation for selected disks. Select disks for write-verify operation with the **dkctl** command. Once set, write-verify defaults appear in the **/etc/default/dkctl** file. The default is **true**. For more information on the disk write-verify feature, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

fsck_ARG
: This parameter provides the arguments to be supplied to the file system checker, **fsck**, when it runs at boot time. The default value is **-xlp**.

reboot_notify_START
: This parameter determines whether or not the system sends notification mail when it reboots. Upon rebooting, the system sends mail to any local users listed in **reboot_notify_ARG**. The default is **false**.

reboot_notify_ARG
: This parameter determines which local logins, if any, receive mail indicating that a system reboot has occurred. To receive notification, you must also set **reboot_notify_START**, above, to **true**. By default, the parameter is set to an empty value.

strtty_START
: This parameter determines whether or not the system pushes STREAMS modules for terminal devices at boot. The default is **true**.

Changing run levels

You must be superuser to change run levels. You change run levels with the **init** command.

Use this command line to go to run level 1, which is administrative mode:

```
# init 1 ↵
```

Enter the following command to take the system to run level 2, which is multiuser mode:

```
# init 2 ↵
```

Enter the following command to take the system to run level 3, which is multiuser mode with network services:

```
# init 3 ↵
```

Changing run levels causes the **init** program to read the **inittab** file looking for all entries containing the specified run level in the *level* field. The entries associate **rc** scripts with run levels. The **init** program then executes all of the **rc** scripts associated with the specified run level. The **rc** scripts perform tasks such as turning on accounting, starting the LP scheduler, and starting various daemons. Output from the **rc** scripts appears in the **/etc/log/init.log** file.

You can also use the **shutdown** command to take the system down to run level S, single-user mode.

Let's follow the sequence that occurs when you invoke **init** to set a run level. Assume that you want to make more services available to your users. 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.
3. The **init** program invokes the `rc.init 3` instruction which is in the *process* field.
4. 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.
5. The **rc.init** program then executes all scripts for run level 3; among these is **syslogd**.

Changing the behavior of and adding rc scripts

You can read the **rc** (run command) scripts to see exactly what they do. We recommend that you do not modify the scripts. You can add your own if needed. For how to add your own **rc** scripts, see *Porting and Developing Applications on the DG/UX™ System*.

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 ONC/NFS)

For example, the **rc.nfsserv** script starts and stops the **nfsd** 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 **nfsd_ARG**. To run twelve copies of the daemon, for instance, set the parameter to:

```
nfsd_ARG="12"
```

System services started at boot

When a system boots, it loads and executes its kernel, the program that provides operating system services. After initializing some of its internal functions and, to some extent, the hardware, the kernel starts the **init** process to start various system services not provided in the kernel itself.

There are a number of system services that the DG/UX system starts when it boots. The services become active at various run levels. These services (not necessarily in order) include:

Package Setup Check
This service checks to see if any software packages on your system are not set up.

Password Check
This service looks for user profiles that do not have passwords.

File System Checker
This service verifies file system integrity.

Local and Remote File Systems
This service makes local and remote (network) file systems available.

Editor Preservation
This service restores editing sessions that may have terminated abnormally.

Batch Job Services
This service manages batch jobs and jobs that run automatically on a regular basis.

System Error Logger
This service handles messages produced by various other system services.

Terminal Lines
This service provides support for terminals and other ports.

Line Printers
This service provides line printer and print queue support.

Accounting
This service accumulates system statistics for accounting purposes. By default, accounting is turned off.

Miscellaneous Service Daemons
This service starts miscellaneous daemons (background processes) that provide a variety of services.

Network Services
These services manage the network software.

You can also define your own services for invocation at system boot time.

DG/UX run levels

A run level is the collection of script-started processes running as a result of an invocation of **init** by the boot process or by a user at the command line. Therefore, for example, run level 2 is the sum of script-started processes invoked by an **init 2** command.

Table 3–1 lists the DG/UX run levels.

Table 3–1 DG/UX run levels

Run level	Description
0	Halts the system.
S or s	Single user mode. This low-level run mode is the default level the system enters upon booting. Only the system default file systems (**/swap**, **/**, and **/usr**) are mounted and available. No processes are running except those of the person logged in as the **root** superuser; essentially the only process running is **init**.
i	Installation mode. All local file systems and disk services are mounted and available, and essential processes are running. The **installman** command is invoked to perform installation tasks. See the **installman** manual page for more information.
1	Administrative mode. This mode is used to install and remove software, and to perform administrative tasks such as checking file systems and doing backups and restores. All local file systems are mounted and available. Only processes associated with the system console can run. You can login only at the console.

Continued

Table 3–1 DG/UX run levels

Run level	Description
2	Multiuser mode. This is the mode with the most service for those who are not operating in a network environment and who are not running the DG/UX X Window System software, Release 4. All local file systems are mounted. Local users can log in at terminals and use local facilities. Some out-bound network services are available. Outside systems cannot contact this system over the network. ONC/NFS services are not available.
3	Multiuser mode. This is the mode required to run DG/UX X Window System software, Release 4 (on workstations, the X Display Manager **xdm** is running). It is also the mode with remote file system sharing (ONC/NFS services are available). Complete multiuser and network services are available.
4	User-defined run level. Used primarily for applications. By default, the same as run level 3.
5	Stops the system and goes to the SCM. This state is equivalent to bringing the system to state **S** and issuing the **halt** command. See the **halt** manual page for more information.
6	Stops and reboots the system using the default boot path. This state is equivalent to bringing the system to state **S** and issuing the **reboot** command. See the **reboot** manual page for more information.
a, b, c	Pseudo run levels. These can be specified without changing a run level.

Normally, you establish a run level from an autoboot path you entered through the SCM Menus or a **sysadm** menu selection. You can, however, change run levels from the shell using the **init** command:

init *run-level* ↵

See the **init** manual page for more information on **init** command arguments.

The services provided at each run level are controlled by the run command (**rc**) scripts 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**, **i**, or one of the numerals **0** through **6**). Therefore, the **init** command, the **inittab** file, and the **rc** scripts together define a run level and determine what processes and services are available on your system.

Run levels are cumulative: the higher the run level, the more services available. Consider the case where you want to make more

services available on a system currently running at single-user mode. You can enter the command **init 2** to change run levels upward from single-user mode S.

You can define the run levels to provide whatever services you choose by adding entries to the **inittab** file using the pseudo run levels a, b, or c. Entering **init a**, for instance, invokes entries in the **inittab** file that have an **a** in the *level* field.

If you configure your system to boot only to run level S or run level 1, you will have to invoke the **init** command yourself to start the desired services. At run level S or run level 1, the system provides some basic services but not all of them. Generally, you get the full complement of services at run level 3, which you reach with this command.

When you boot the DG/UX system to run level 2 or 3, which are the multiuser states:

- 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, the batch job scheduler, various disk-related services, the network status monitor, and the network lock daemon are started.
- The LP system and UUCP are ready to use. The Service Access Facility (SAF) monitors ports, providing whatever services you have configured it to provide. SAF's most notable service is monitoring user terminal lines and starting the **login** program for users who want to log in.
- If used, TCP/IP transmissions work outward in run level 2, and work in both directions in run levels 3 and 4.

OS clients and OS servers operate at run level 3, since network services are required to support OS clients.

rc scripts started for each run level

For systems with the TCP/IP and ONC/NFS packages (in addition to the DG/UX system) loaded and set up, Table 3–2 shows which **rc** scripts are started per run level. Note the cumulative effect: the higher the run level, the more processes are running. Blanks indicate that a script is not running.

Table 3–2 rc scripts started for each run level

i	S	0	1	2	3	4	5	6
	rc.ups	rc.ups	rc.ups	rc.ups	rc.ups	rc.ups		
			rc.tcload	rc.tcload	rc.tcload	rc.tcload		

Continued

Table 3–2 rc scripts started for each run level

i	S	0	1	2	3	4	5	6
rc.update			rc.update	rc.update	rc.update	rc.update		
rc.localfs			rc.localfs	rc.localfs	rc.localfs	rc.localfs		
			rc.sync	rc.sync	rc.sync	rc.sync		
			rc.lan	rc.lan	rc.lan	rc.lan		
			rc.setup	rc.setup	rc.setup	rc.setup		
			rc.daemon	rc.daemon	rc.daemon	rc.daemon		
rc.install								
				rc.usrproc	rc.usrproc	rc.usrproc		
				rc.llc	rc.llc	rc.llc		
				rc.syslogd	rc.syslogd	rc.syslogd		
				rc.dgserv	rc.dgserv	rc.dgserv		
				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.failover	rc.failover		
		rc.halt					rc.halt	
								rc.reboot
				rc.tcpip-port	rc.tcpip-port	rc.tcpip-port		
					rc.tcpip-serv	rc.tcpip-serv		
				rc.ypserv	rc.ypserv	rc.ypserv		
				rc.nfslockd	rc.nfslockd	rc.nfslockd		
					rc.nfsserv	rc.nfsserv		
					rc.nfsfs	rc.nfsfs		

For an explanation of each **rc** script, see "rc scripts."

Files and scripts that control booting to specific run levels

The **inittab** file contains entries specifying which processes will be invoked at which run level. The **init** program reads the entries in **inittab**. When an **inittab** entry matches the specified run level, **init** passes it to a shell for execution.

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 scripts in the **rc*N*.d** directories are run command (**rc**) scripts. The **rc** scripts start and stop system services required by run levels S through 6. Output from **rc** scripts goes to **/etc/log/init.log**.

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 **rc** script in **init.d**.

The **init.d** directory contains all of the **rc** 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.

When **rc.init** invokes a run level, the characteristics of that run level are produced by the execution of specified **rc** and check scripts in **/usr/sbin/init.d**.

rc scripts

The **rc** scripts are located in **/usr/sbin/init.d** and all have the prefix **rc**. These scripts are invoked with either a start or stop argument.

The **rc** scripts shipped with the DG/UX system and their functions are:

rc.ups Starts the UPS (uninterruptible power supply) daemon (for systems with the UPS subsystem hardware only). This script runs in single-user mode and in levels 0 through 4.

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

rc.update Starts various disk-related services in run levels i, 1, 2, 3, and 4. These services include the block I/O daemon (**biod**) and the write-verify service.

rc.localfs Mounts local file systems listed in **/etc/fstab** in run levels i, 1, 2, 3, and 4; unmounts them in all other run levels. A local file system is one of type **dg/ux**, **ramdisk**, **dos**, or **cdrom**. The value of the **localfs_ARG** variable, set in **/etc/dgux.params**, determines which file system types to mount.

rc.sync	Loads the synchronous controllers used for wide-area network (WAN) communication. This script runs at run levels 1 through 4.
rc.lan	Loads the controllers used for local-area network (LAN) communication. This script runs at run levels 1 through 4.
rc.setup	Displays packages that have not been set up at initial boot.
rc.daemon	Starts miscellaneous daemons.
rc.install	Performs installation of the DG/UX system. This script runs only at run level i.
rc.links	Create, list, or remove links in the **/etc/rc?.d** directories, where **?** is a regular expression pattern-matching metacharacter. This runs only when you set up the DG/UX system (not during a regular change of run level). You can, if you wish, use **rc.links** to create, list, or remove your own links. This file is a binary executable rather than a shell script.
rc.usrproc	Kills all user processes in run levels S, 0, 1, 5, and 6.
rc.llc	Starts the llc daemon (**llcd**), which provides logical link control services in run levels 2, 3, and 4.
rc.syslogd	Starts the **syslog** error logging program in run levels 2, 3, and 4; kills it in all other run levels.
rc.dgserv	Starts DG/UX system services in run levels 2, 3, and 4. This script starts the **dgsvcd** daemon, which provides services for the AV/ALERT facility.
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** command in run levels 2, 3, and 4 to recover editor files saved during a system crash.
rc.failover	Starts **failoverd**(1M) for communicating with another host used for failover disks. This script runs at levels 3 and 4.

rc.halt Halts the processor, taking it to the SCM. This script runs at run levels 0 and 5.

rc.reboot Halts and reboots the system. This script runs only at run level 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.tcpipserv In run levels 3 and 4, starts whichever TCP/IP daemons are defined to run on your system. The TCP/IP daemons are **telnetd**, **ftpd**, **tftpd**, **smtpd**, **rlogind**, **rwhod**, **rshd**, and **rexecd**. The **rc.tcpipserv** script kills them in all other run levels.

rc.ypserv Starts the **yp** and **portmap** daemons, and sets the domain name in run levels 2, 3, and 4; kills these in all other run levels.

rc.nfslockd Starts daemons for ONC/NFS file locking, **statd** and **lockd**.

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.nfsfs Mounts all local and ONC/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 **rc** scripts, the DG/UX system uses check scripts to set up a properly running environment. Check scripts are stored in **/usr/sbin/init.d** / and all have the prefix chk.

Check scripts are usually run once, 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 shipped with the DG/UX system and their functions are:

chk.date Displays the current system date and allows you 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 **-xlp** switch to check file systems in parallel, checking only those file systems that need checking.

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

- Initializes the **/etc/ps_data** file.
- Cleans out 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 in which 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/devlinktab** file.

chk.strtty This script initializes terminal ports by pushing the required STREAMS modules. The script initializes **duart** and **syac** lines and pseudo-terminals.

init.d links that invoke rc scripts

Typically, **rc** and check scripts exist in **/usr/sbin/init.d**. The **rc** scripts are invoked via links in an **/etc/rc*N*.d** directory. There are nine **/etc/rc*N*.d** directories: **/etc/rcS.d**, **/etc/rc0.d**, **/etc/rci.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`	Defines whether the process should be started (`S`) or killed (`K`) upon entering the new run level.
nnn	A number from 000 to 999 indicating 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 script name in **/usr/sbin/init.d**.

All process scripts are specified to be either killed or started when you change run levels. The **rc.init** 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. All scripts in **init.d** have links in all **/etc/rc*N*.d** directories; the **K** or **S** prefix determines 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 ↵
K237.ypserv    S116.sync      S212.llc       S239.nfslockd
S315.nfsserv S015.ups       S117.lan       S232.tcpipport
S251.account    S334.tcpipserv S112.tcload    S119.setup
S235.syslogd   S252.cron       S353.nfsfs S113.update
S130.daemon    S236.dgserv    S253.lpsched    S358.failover
S114.localfs   S210.usrproc   S237.ypserv    S254.preserve
```

The complete layout of how all **rc** scripts are started and killed is in **/etc/dgux.rclinktab.proto**. Figure 3–1 shows a few lines from that file:

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

    rc.ups               015  S   S   S   S   S   S   K   K   -

##  rc.usrfs             111  K   K   S   S   S   S   K   K   -
    rc.tcload            112  K   K   S   S   S   S   K   K   -
    rc.daemon            130  K   K   S   S   S   S   K   K   -
    rc.install           131  -   -   -   -   -   -   -   -   S

    rc.halt              511  -   S   -   -   -   -   S   -   -

    rc.reboot            611  -   -   -   -   -   -   -   S   -
```

Figure 3–1 Lines from the /etc/dgux.rclinktab.proto file

You can think of all **rc** scripts as being either on (S) or off (K). The **/etc/dgux.rclinktab.proto** file contains comments explaining its contents.

The /etc/inittab file

The **init** program relies on the information in the **/etc/inittab** file. Entries in the **inittab** file have this format:

id:level:action:process

where the fields are as follows:

id
One, two, or three characters that uniquely identify an entry.

level
A character (`s`, `0` through `6`, `a`, `b`, or `c`) that determines at what run level the specified action is to take place. If the level field is empty, the action occurs at all run levels.

action
One of the following:

`boot` Process the entry the first time **init** leaves single-user mode. Do not wait for the process to terminate.

`bootwait` Process the entry the first time **init** goes from single-user mode to multi-user mode after the system is booted. If **initdefault** is set to 2, the entry is processed at boot time. **init** starts the process, waits for its termination and, when it dies, does not restart the process.

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

`off` At the specified level, kill the process or ignore it.

`once` Run the specified process once and don't start it again if it finishes.

`ondemand` Synonymous with `respawn`, but used only when the level is `a`, `b`, or `c`.

`powerfail` Execute the process in this entry only when **init** receives a power fail signal (SIGPWR). See **signal**(2).

powerwait Execute the process in this entry only when **init** receives a power fail signal and wait until it terminates before continuing any processing of **inittab**.

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

sysinit Process the entry before **init** attempts to access the system console. Wait for the process to terminate before continuing.

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

process
Any executable program, including shell procedures.

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

A copy of the prototype **inittab** file and a discussion of its contents follows:

```
#
def:s:initdefault:
ttc::sysinit:/sbin/autocon                </dev/console >/dev/console 2>&1
fsc::bootwait:/sbin/chk.fsck              </dev/console >/dev/console 2>&1
dat::bootwait:/usr/sbin/init.d/chk.date   </dev/console >/dev/console 2>&1
set::bootwait:/usr/sbin/init.d/chk.system </dev/console >/dev/console 2>&1
tty::bootwait:/usr/sbin/init.d/chk.strtty </dev/console >/dev/console 2>&1
dev::bootwait:/usr/sbin/init.d/chk.devlink </dev/console >/dev/console 2>&1
#
rc0:0:wait:/sbin/rc.init 0 >/dev/console 2>&1
rci:i:wait:/sbin/rc.init i >/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
#
# ttymon is more secure than su since su is always on console
con::respawn:/usr/lib/saf/ttymon -g -p "Console Login: " -d /dev/console \
       -l console
sec::off:#/sbin/su - 1 </dev/console  >/dev/console 2>&1
#
saf:234:respawn:/usr/lib/saf/sac -t 45  #Service Access Facility
```

```
# When init receives a SIGPWR, it kicks itself with 'init S':
ups::powerfail:/sbin/init S < /dev/console > /dev/console 2>&1
```

Figure 3–2 The prototype /etc/inittab file

Note: The line beginning with `con` was broken for readability.

The first line in the file sets `s`, single-user mode, as the default initialization run level.

The next line makes the system console usable by pushing the required STREAMS modules (for line discipline and so on) onto the stack that controls the system console.

The next five lines start up five check scripts: **chk.fsck, chk.date, chk.system, chk.strtty, and chk.devlink**. These scripts are executed at boot time according to the **bootwait** action of **inittab**.

The next eight lines are instructions for setting run level **i** (installation) and 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.

After the run level lines, the line `con` identifies the operator's console (**/dev/console**) to the system. The line after it, `sec`, is an alternate (less secure) service to run on the system console. Note that this line is turned off. The next line, `saf`, starts the Service Access Facility (SAF), at run levels 2, 3, and 4 to provide terminal services to users. The last line, `ups`, shuts down the system if a powerfail condition occurs (this service exists only on systems with the uninterruptible power supply (UPS) subsystem).

Handling boot time problems

During system boot, you may be alerted to a class of recoverable operating problems through status codes. These codes are presented on the system console and are logged to the **/usr/adm/messages** file. The **/usr/release/dgux_5.4R3.10.status.codes** file contains status message numbers and a brief description of each detected problem. An example follows:

```
IO_EIO_DEVICE_TIMED_OUT                0x100a18 04005030
```

To find the file containing information on status code `04005030`, you would execute the following commands:

```
# cd /usr/release ↵
# more dgux_5.4R3.00.status.codes ↵
```

Then search for the desired code in the status code file using this command:

```
/5030
```

You then read the indicated file using a command such as **view** or **more**, both of which offer commands for locating the desired text.

```
IO_EIO_DEVICE_TIMED_OUT                0x100a18 04005030
    Device did not respond within timeout period.
```

Most of these messages relate to minor hardware problems, that, once corrected, allow continued normal operation. For example, you could receive a message when attempting to access a file that is located in a disk drive that is turned off. In most cases, the problem is recoverable. However, other conditions may eventually lead to a hang or a panic.

In other cases, the system warns of problems that you should investigate. For example:

```
SCI_EINTR_NFS_SERVER_PROCESS        0x6c0814 33004024
   An NFS daemon process was terminated by a signal.
```

This message indicates that network services have been suspended. You need to troubleshoot the problem to restore NFS to proper operation.

End of Chapter

4 DG/UX files, directories, and swap areas

This chapter lists and describes some of the files and directories shipped with and created by the DG/UX system. 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 page.

The chapter also explains how to:

- View, edit, and find files
- Controlling the size of files and directories
- Clean out temporary file directories and log files
- Manage swap areas

WARNING: You should modify files or directories supplied with the DG/UX system only if you know the effect of the modification. You also should not change any of the system-supplied directories to symbolic links. Modifying these files and directories may prevent you from upgrading your system to a new revision of DG/UX.

Directory structure and navigation

The DG/UX system directory structure is like an inverted tree in that branches grow from a central root. The root is the directory /, called the root directory. The root directory contains files (such as the kernel, **/dgux**) and directories (such as **/tmp**).

In general, any directory can contain files and other directories. Some directories, called mount point directories, are equivalent to the file systems defined on your system. For detailed information about file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

The pathname of a file or directory represents the hierarchy of directory names that leads from the root directory to the given file or directory. For example, the pathname of the **cat** command is **/usr/bin/cat**, indicating that the **cat** program (a file) resides in the **bin** directory, which resides in the **usr** directory, which resides in the / (root) directory. You may specify any file or directory uniquely by using its entire pathname.

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).

Every process running on the system, including your shell, has a current directory. The significance of a current directory is that it allows the process to run (or execute) a file or directory without a full pathname, using a name relative to its current directory. Commands to run a file or directory specified using relative path-naming are successful only if the current directory is specified in your search path (a period (.) — signifying the home directory — must be in the path). See *Using the DG/UX™ System* for information on setting the PATH variable.

Changing from one current directory to another (cd)

To change your current directory, use the **cd** command:

```
# cd  /  ↵
```

The preceding command puts you in the root directory. You could have also changed to the root directory from **/admin** by using this form of the **cd** command:

```
# cd  ..  ↵
```

The **..** (dot-dot) notation refers to the next highest directory, regardless of your current position in the file system structure. To return to the **/admin** directory from the root directory, use this command:

```
# cd  admin ↵
```

To change from the **/admin** directory to the **/usr** directory, you could use a **cd** command specifying an absolute pathname, like this:

```
# cd  /usr ↵
```

Or you could use a **cd** command specifying a relative pathname, like this:

```
# cd  .. /usr ↵
```

To return to your home directory (where you were when you first logged into the system), use the **cd** command without an argument:

```
# cd  ↵
```

Finding out your current directory (pwd)

To see your current directory, use the **pwd** command:

```
# pwd ↵
```

Seeing the contents of a directory (ls)

Once in a directory, list the directory's contents with the **ls** command. The **ls** command has a number of options, of which two commonly used ones are:

-a This option lists all files in a directory. Normally, the **ls** command lists all files except those beginning with . (period). Files starting with dots are typically personal configuration files, such as those in your home directory that initialize your environment. The **.profile** file in the **/admin** directory is an example of such a file.

-l This option produces a long listing. By default, the **ls** command lists only the names of files and directories. The long listing includes several attributes of the listed file or directory.

An example of running the **ls** command follows:

```
# ls -l chap1 ↵
-rw-r--r--   1 smith   doc  22319 Sep 25 10:32 chap1
```

Each part of the output line is explained as follows:

`-rw-r--r--` The mode is a series of hyphens or letters indicating the type of file and the permissions. If the first character is **d**, the file is a directory. If the first character is -, the file is a regular file. This example shows a regular file. There are letters other than **d**.

The next nine letters indicate permissions: the first three are the owner permissions, the next three are the group permissions, and the last three are permissions for other users.The letter **r** represents read access, **w** represents write access, and **x** represents execute access for executable files (such as programs) and search access for directories. The symbol - indicates that the permission is denied. The **rwx** - letters are a subset of the full set of permissions. See the **ls**(1) manual page for a complete list. You change the permissions of a file with the **chmod** command.

`1`	The link count is the number of physical pointers linked to that file.
`smith`	This field indicates the username of the file's owner. When you create a file or directory, the system attaches your name to the file as the owner. You change the ownership of a file with the **chown** command.
`doc`	This field indicates the name of a user group for which you may assign special permissions. Group permissions appear in the mode of the file or directory. When you create a file or directory, the system attaches your group to the file. You change the group of a file with the **chgrp** command.
`22319`	This is the size of the file in bytes. For directories, this value represents the size of the directory data itself, not the contents of the directory.
`Sep 5 10:32`	The date and time stamp tells when the file was last modified. If the file has not been modified since it was created, the date and time stamp indicate creation date and time.
`chap1`	This is the name of the file or directory.

See *Using the DG/UX™ System* and the **ls** manual page for more information on the **ls** command.

Contents of the root directory

The **root** logical disk is mounted on the directory /. We refer to directories and files on the **root** logical disk as "being in root." The **root** directory contains files and directories that are relevant to the host on which they are stored and that are necessary to boot the system.

/.profile

The **/.profile** file is the environment initialization and configuration file for the **root** login shell. We recommend that you use the **sysadm** login rather than the **root** login for administrative work.

/admin

The **/admin** directory is the home directory for the **sysadm** login account. We recommend that you use the **sysadm** login account for administrative work rather than the **root** login so that your work files will be in **/admin** rather than /.

By default, the **/admin** directory contains a prototype shell initialization file, **.profile.proto**, and a working copy of the file, **.profile**.

The **/admin** directory also contains the **crontabs** directory, a repository for prototype **crontab** files useful for routine system management. The **/admin/crontabs** directory contains the following files:

Table 4–1 Prototype crontab files shipped with DG/UX

File	Description
acct.proto	Contains jobs for running the accounting system.
root.proto	Contains jobs for maintaining the system. These jobs perform a variety of functions such as cleaning out temporary file directories and truncating logs.
lp.proto	Contains jobs for maintaining the LP subsystem.
uucp.proto	Performs functions for the UUCP file transfer/remote command execution utility.

Chapter 6 discusses the **cron** facility and the prototype **crontab** files in more detail.

IMPORTANT: If you need to add to or modify the contents of a prototype file, make a copy of the file, using the same name but without the **.proto** suffix, and modify the file copy. Upgrades of the DG/UX system overload prototype files (files with the **.proto** suffix), but not the file copies without the suffix. Therefore modifying a copy of a prototype file preserves your changes during a DG/UX system upgrade.

/bin

/bin is a symbolic link to the **usr/bin** directory, which contains public commands.

/dev

The **/dev** directory contains device nodes, also called special files.

/dgux

The **/dgux** file is the executable kernel file.

/dgux.aviion

The **/dgux.aviion** file is the default name of the kernel executable file and is typically a hard link to the **dgux** file.

/dgux.installer

The **/dgux.installer** file is the executable kernel used for installation of system software.

/etc

The **/etc** directory contains configuration files and system data. For more information about the contents of the **/etc** directory, see "Contents of the /etc directory" in this chapter.

/lib

/lib is a symbolic link to the **usr/lib** directory that contains object libraries.

/local

The **/local** directory is a repository for site-specific files.

/opt

The **/opt** directory is the parent directory for user-configurable portions of applications packages.

/proc

The **/proc** directory is currently unused.

/sbin

The **/sbin** directory contains minimum system commands to get the system up.

/srv

The **/srv** directory is a mount point for the **srv** logical disk that contains client and release management files. For more information about the contents of the **/srv** directory, see "Contents of the /srv directory" in this chapter.

/tftpboot

The **/tftpboot** directory, used only on OS servers and X terminal servers, contains links to OS and X terminal clients' bootable executable files in the **/usr/stand** directory.

/tmp

The **/tmp** directory storages temporary files at the request of system and application processes.

/usr

The **/usr** directory is the mount point directory for the **/usr** file system. For more information on the contents of this directory, see "Contents of the /usr directory" in this chapter.

/var

The **/var** directory stores system data files whose sizes vary as the system runs. For more information about the contents of this directory, see "Contents of the /var directory."

Contents of the /etc directory

The **/etc** directory contains files that you and your system management tools alter to customize your system.

Technically, you can alter most of these files yourself using a text editor. However, we recommend that, in cases where **sysadm** or another administrative utility offers an interface for managing a file, you use the interface to alter the file rather than an editor. The rationale for this recommendation is twofold:

- An interface program such as **sysadm** will not corrupt the file it maintains, whereas there is some risk that you may accidentally corrupt the data in the file if you alter it with a text editor.
- An interface program continues to provide the desired service in future revisions of the software, even if such revisions involve changes such as moving the data file, changing its format, or re-implementing the related service in a completely different manner.

Many of the files in **/etc** and other system directories have *prototypes*, files representing the state of the file as it was shipped with the software release. For example, the prototype for the **passwd** file is **passwd.proto**. You may find the prototype files useful if you wish to restore or refer to the default configuration of the system. For a complete list of prototype files shipped with the DG/UX system in the **/etc** directory, see the **/etc/dgux.prototab** file.

If you need to add to or modify the contents of a prototype file, make a copy of the file, using the same name but without the **.proto** suffix, and modify the file copy. Upgrades of the DG/UX system overload prototype files (files with the **.proto** suffix), but not the file copies without the suffix. Therefore modifying a copy of a prototype file preserves your changes during a DG/UX system upgrade.

/etc database files maintained by the system

This section describes some of the files created or maintained by system services other than the **sysadm** utility.

/etc/devlinktab — This file contains entries used to make short-named links to device nodes with otherwise unwieldy names. The system adds to this file at boot to reflect the current configuration of the system. Note that the system does not rebuild this file. Entries are never deleted from **/etc/devlinktab**. An example follows:

```
/dev/rmt          0          st(insc@7(FFF8A000),4,0)
```

The entry above indicates that the tape device with SCSI ID 4 on the integrated SCSI bus appears as device **/dev/rmt/0** in the file system.

/etc/dgux.rclinktab — This data table determines how the **rc.links** script creates or removes the links in the **rc*.d** directories that point to files in the **/usr/sbin/init.d** directory. The DG/UX system and other software packages modify this file during package setup.

/etc/log — A directory containing logs for various system services, including information on run level changes and daemon activity. The system generally creates these logs during the boot process. These logs are useful for reviewing activity that occurs during the boot process and changes of run level.

/etc/rc*.d — There are nine of these directories in **/etc**: **rc0.d**, **rc1.d**, **rc2.d**, **rc3.d**, **rc4.d**, **rc5.d**, **rc6.d**, **rcS.d**, and **rci.d**. Each directory contains links to all the shell scripts in **/etc/init.d**. Software packages modify these directories during setup, reflecting any such changes in **/etc/dgux.rclinktab**. See Chapter 3 for a complete list of all services that can be set for each run level.

/etc/utmp — The **/etc/utmp** file contains information on the run level of the system. Various system services, such as **login**, maintain this information. Use the **who** command to access this information.

/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. This file contains a record for each login that occurs. Use the **last** and **who** commands to access this file. Periodically, this file should be cleared or truncated. For how to truncate the **/etc/wtmp** file, see "Truncating the /etc/wtmp file."

/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/dgux.params — This file contains parameters that you can set to control the actions of **rc** scripts in **/etc/init.d**. The **rc** scripts determine what happens during boots and other changes of run level. Chapter 3 discusses run levels.

/etc/dumptab — This file contains the dump table which lists the different media supported by **dump2**. It describes the media characteristics for each medium made available to **dump2**. See **dumptab** and Chapter 8 for more information on system backups.

/etc/failover — This directory contains the databases for failover disks. You maintain these databases using the operations in the **sysadm** menu `Device-> Disk-> Failover`. For more information on failover disks, see *Achieving High Availability on AViiON® Systems*.

/etc/fstab — The **fstab** file specifies the file systems to be mounted by the **/etc/mount** command. The following is a sample entry in **fstab**. Note that it is in NFS format; we recommend this format even if you are not using 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 NFS file systems as well as local file systems. The old style **fstab** entries are also supported. See **fstab** for detailed information.

/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 separated by colons (:):

```
group_name:password:group_id:login_names
```

For information about groups, see Chapter 5 and the **group** man page. If you have ONC/NIS, see the **yppasswdd** man page and *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

/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/lp — This directory contains files supporting your configuration of the LP subsystem. For more information on LP, see *Installing and Managing Printers on the DG/UX™ System*.

/etc/nfs.params — This file contains parameters for controlling ONC/NIS and NFS services.

/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
```

For more information about passwords and this file, see Chapter 5 and **passwd**.

/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. Also see your TCP/IP documentation.

/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** operation System–> Date–> Set. The TZ variable can be redefined on a user (login) basis by setting the variable in the associated **.profile** or **.login** file.

/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/cron.d — This directory contains files that you modify to customize your system's **cron** services. The files here include **at.allow** and **cron.allow**, which list users who are permitted to submit jobs for execution by **cron**, **at**, and **batch**. The **queuedefs** file, described in more detail in **cron**, contains parameters governing the **at** and **batch** queues. See Chapter 6 and the **crontab** manual page, in addition to the manual pages mentioned above, for more information.

/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; run level 2 is multiuser mode; and run level 3 multiuser mode with network services. Some applications packages, such as X11, modify this file during package setup. Chapter 3 summarizes the various run levels and describes their uses. See the **inittab** manual page for more information.

/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 commands in these global profile files. For more information, see Chapter 5 and *Using the DG/UX™ System*

/etc/motd — The **/etc/motd** file contains the message of the day. The system shell initialization files, **login.csh** and **profile**, print this message to users logging in.

/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 commands in these global profile files. See Chapter 5 for more information.

Administrative commands in the /sbin directory

The following commands are available in **/sbin**. These are the minimum system administration commands necessary to operate the system.

/sbin/chk.fsck — An **rc** check script that invokes **fsck** during boot.

/sbin/fsck — Runs the **fsck** file system check program.

/sbin/halt — Halts the operating system and restores control to the firmware-based hardware control system, the SCM.

/sbin/init — Command to change run levels S, 1, 2, 3.

/sbin/mount — Mounts a file system on the DG/UX directory tree.

/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
```

/sbin/setup.d/boot — Sets up scripts that must run on the host system CPU.

/sbin/setup.d/root — Contains scripts that set up software packages in the / file system.

/sbin/sh — The DG/UX **sh** command.

/sbin/shutdown — Brings the operating system down to single-user mode (run level S). For information on run levels, see Chapter 3.

/sbin/su — Switches user (login) name. For instance, **su sysadm** changes your user ID to **sysadm.**

/sbin/ttymon — Enables login on the console line and other terminal lines.

/sbin/umount — Unmounts a remote file system.

Contents of the /srv directory

The **/srv** directory contains the directories and files needed for managing operating system releases and clients. Table 4–2 lists the files and directories in the **/srv** directory.

Table 4–2 Contents of the /srv directory

Name	Description
/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.

Continued

Name	Description
/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.

Contents of the /usr directory

The **usr** logical disk is mounted on the **/usr** directory. Files in this directory are release-dependent and generally read-only. The physical and logical directories and the files on the **usr** logical disk are listed in Table 4–3.

Table 4–3 Contents of the /usr directory

Name	Description
/usr/adm	Symbolic link to **/var/adm**.
/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** command.
/usr/etc	Contains database and configuration files.

Continued

Name	Description
/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 ONC/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 16.
/usr/lib/gcc	Symbolic link to **/usr/lib/gcc-2**.
/usr/lib/gcc-2	Contains the DG/UX GNU C compiler. For information on this compiler, see the Release Notice that comes with the compiler package.
/usr/local	Contains site-specific, read-only files.
/usr/mail	Symbolic link to **/var/mail**.
/usr/news	Symbolic link to **/var/news**.
/usr/opt	Contains applications packages.
/usr/preserve	Symbolic link to **/var/preserve**.
/usr/release	Contains media notices, release notices, and system package names.
/usr/root.proto	Prototype / (root) file system copied when you add an OS client.
/usr/sbin	Commands used only by a superuser managing the system.
/usr/sbin/init.d	Contains executable files (the check scripts and **rc** scripts) used to change the system run level. 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.

Continued

Name	Description
/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 ONC/NFS.
/usr/share	Contains release independent shared software.
/usr/spool	Symbolic link to **/var/spool**.
/usr/src	Parent directory for source code.
/usr/src/uts/aviion/cf/system.*.proto	Contains your custom version of the devices and configuration parameters listed in **/usr/etc/master.d/dgux**. For configuration, the **config** 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**.
/usr/src/uts/aviion/lb	Contains the kernel libraries which are used to build the kernel image. See Chapter 15. Also see **config** and **make**.
/usr/stand	Contains stand-alone utilities and bootstrap programs.
/usr/tmp	Symbolic link to **/var/tmp**.
/usr/ucb	Symbolic link to **/usr/bin**.

Contents of the /var directory

The **/var** directory contains files that are release-dependent, host-dependent, have read and write permissions set, and are sized dynamically. Table 4–4 lists the directories and files in the **/var** directory.

Table 4–4 Contents of the /var directory

Name	Description
/var/adm	This directory contains a variety of files produced by the system, such as system error logger (**syslogd**) messages. This directory also contains the data collection files for the accounting system. See Chapter 16 for information on accounting system files and directories.
/var/adm/acct	Directory containing the data necessary to provide accounting reports.
/var/adm/log	Contains log files produced by system services.
/var/adm/ messages*	Contains information about the system logged by **syslog.d**. The files **messages.0**, **messages.1**, and **messages.2** are created by a **cron** job supplied in **/admin/crontabs/root.proto**.
/var/adm/pact	A file used by the accounting software.
/var/adm/ spellhist	Contains words considered misspelled by the **spell** utility.
/var/adm/sulog	Contains a history of superuser (**su**) command usage. As a security measure, this file should note be readable by others.
/var/Build	Kernel builds by **sysadm** Auto Configure and Build operations take place here.
/var/cron	Contains the **cron** log.
/var/cron/log	Contains a history of all actions taken by **/usr/sbin/cron**.
/var/ftp	The home directory for **ftp** users. Contains utilities for **ftp** users.
/var/lp	Contains logs for the LP print service.
/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/saf	Contains logs for the Service Access Facility.

Continued

Name	Description
/var/spool	Contains spooling files for LP, UUCP, and **cron**.
/var/spool/lp	Contains all of the files and directories of the LP system.
/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.
/var/ups	Contains logs and the status file for the Uninterruptible Power Supply **upsd** daemon.

File permissions

On the DG/UX system, a file's owner controls read, write, and execute access to the file. Access permissions in DG/UX are controlled as in most other UNIX implementations. For a detailed discussion of permissions, see *Using the DG/UX™ System*.

Viewing files

There are several commands that let you view the contents of a file.

Viewing a short file (cat)

For short files, such as the **sysadm** profile's shell initialization file, **.profile**, use the **cat** command:

```
cat  /admin/.profile  ↵
```

The **cat** command prints the entire file to the screen without stopping.

Viewing a long file one screen at a time (more)

For longer files, such as the DG/UX system release notice, use the **more** command:

```
more  /usr/release/dgux_5.4.3.rn  ↵
```

The **more** command prints text to your display one screen at a time, allowing you to advance to the next screen by pressing the space bar. To exit **more** before you reach the end of the file, press **q**. To list other commands while using **more**, press **h** or ?.

The **more** command is good for files that contain special attributes such as highlighting. The **more** command does not, however, allow you to navigate through a file as easily as the **view** command.

Viewing and navigating a long file (vi and view)

You can view a file by opening it inside of the **vi** editor shipped as part of the DG/UX system. The **view** command is basically the DG/UX editor **vi** started with a file in read-only mode. With **view**, you can only read and cannot change a file. So **vi** enables you to look at a file and change it if your user login has write permission on that file, while the **view** command only allows you to view and move around a file.

To start the **view** command, enter a line like the following:

```
# view  /usr/release/dgux_5.4.3.rn  ↵
```

To start the **vi** command, enter a line like the following:

```
# vi  /usr/release/dgux_5.4.3.rn  ↵
```

Table 4–5 shows commands you use to move through a file opened in **view** and **vi**.

Table 4–5 Navigation commands for vi and view

Command	Description
<Ctrl–F>	Move forward one screen.
<Ctrl–B>	Move backward one screen.
j	Move forward one line.
25j	Move forward 25 lines.
k	Move backward one line.
0	Go to the beginning of the line.
$	Go to the end of the line.
l	Move forward one character.
h	Move backward one character.
w	Move forward one word.
b	Move backward one word.
/string	Search forward for string.
?*string*	Search backward for string.
G	Move to the end of the file.
:1	Move to the first line of the file.
<Esc>	Return to command mode.
:q	Quit (exit) the file.
:q!	Quit (exit) the file without saving changes.

You can specify numeric arguments before a command to repeat it a number of times, as demonstrated by the **25j** command. The searching commands, / and ?, and the last-line mode commands, that start with **:**, move the cursor temporarily to the bottom of the screen. Pressing Enter executes the command and returns the cursor to the text area of the screen.

Editing a file (vi)

The **vi** editor operates in two modes: command mode and input mode. When you first invoke **vi**, it is in command mode. In command mode, you can navigate through the file as shown in Table 4–5 and issue editing commands. Some editing commands perform functions like deleting or copying a character, word, line, and so on. Others place you in input mode where you can type text into the file. To exit from input mode and return to command mode, press **<Esc>**.

Table 4–6 shows some **vi** commands for changing a file.

Table 4–6 Basic vi editing commands

Keystroke	Description
i	Enter input mode, inserting text at current cursor position.
I	Enter input mode, inserting text at the beginning of the line.
A	Enter input mode, appending text to the end of the line.
o	Enter input mode,opening a line below the current line and moving text in it.
O	Enter input mode, opening a line before the current line and entering text in it.
dd	Delete the current line.
dw	Delete to the end of the current word.
x	Delete the character under the cursor.
yy	Copy ("yank") the current line.
p	Insert ("put") the most recently yanked or deleted line below the current line.
P	Insert ("put") the most recently yanked or deleted line before the current line.
<Esc>	Exit input mode and return to command mode.
:w	Write changes to the file.
:w file2	Write changes to a file called **file2**.
:wq	Write changes to the file and exit ("quit") the file.
:q!	Exit ("quit") the file without saving changes.

For more information on the **view** command, see the **view** manual page. For more information on the **vi** editor, see the **vi** manual page and *Using the DG/UX™ System Editors*.

Finding files and directories

Select the **sysadm** operation `File System-> File Information-> Find` operation to search a specified directory nand list all files or directories under it that satisfy specified criteria. The operation can also sort the output data in various ways. (In this discussion, the term *file* also refers to directories.)

The Find operation can help you find:

- Files whose names match a specific pattern
- Files belonging to particular users or groups
- Files of a given type
- Files that have not been accessed or modified in a long time and are no longer necessary

- Files that take up too much space

You can also determine the number and order of files found. You can sort the information by name, size, access date, and modification date.

The Find operation presents you with the following queries:

`Directories`
List the directories you want to search. The operation searches the named directories as well as any directories underneath them. If you specify no directories, the operation searches the entire system.

`Restrict to Local File Systems`
Select this option to restrict the search to file systems that reside physically on your system. Without this option, the operation searches local directories as well as file systems mounted from remote systems.

`Restrict to This File System`
Select this option to restrict the search to the file systems containing the named directories. Without this option, the operation searches all directories under the named directories as well as any file systems mounted under the directories.

`File Name`
Specify name or name pattern to match. You may use the metacharacters (wildcards) accepted by **sh**. These metacharacters are:

?
Matches any one character.

*
Matches any number of any characters.

[ ]
When surrounding a group of characters (not including the hyphen,–), matches any one character in the group. For example, **[abc]** matches an occurrence of **a**, **b**, or **c**. Use the hyphen to represent a range of characters. For example, **a–z** represents any lowercase letter. Follow the closing bracket with **!** to negate the set, causing the expression to match any character *not* in the set. For example, **[ag–iA12]!** matches any character except **a**, **g**, **h**, **i**, **A**, **1**, or **2**.

If the name contains metacharacters, surround it with quotation marks. For example, the pattern "**c*.[1–4ab]**" matches any file or directory name starting with **c** and ending with a dot followed by one of the characters **1**, **2**, **3**, **4**, **a**, or **b.**

`Owner Name or ID`
Specify the login name or user ID number of an existing user. The operation finds only files that the user owns. Remember that usernames are case sensitive.

`Group Name or ID`
Specify the group name or group ID number of an existing group. The operation finds only files whose group ownership is for the specified group. Like usernames, group names are case sensitive.

`File Type`
Specify the type of file to find. You may specify more than one type. The DG/UX file system and the Find operation recognize these file types:

`any`
Any of the following types.

`regular`
A typical file such as a text file that is not a directory or any other type specified here.

`directory`
A directory.

`block special`
A device file created for block data access.

`character special`
A device file created for character (raw) data access.

`fifo (named pipe)`
A named pipe.

`Days Since Last Modification`
When a user modifies a file, the system records the date in the file's inode, the block containing data about the file. The Find operation can use this information to search for files that have not been modified since a particular date.

`Days Since Last Access`
A file's inode also includes the date the file was last accessed (read). The Find operation can use this information to search for files that have not been accessed since a particular date.

`File Size (bytes)`
Specify a size limit for files. The operation searches for files larger than this size.

`File Sorting Method`
The operation lets you organize the data about files that it finds. You may choose to sort files by name, size, access time, or modification time. You can also specify increasing order or decreasing order for the sort, or you can specify no sorting at all.

`Maximum Number of Files to Report`
To limit the number of files in the display, specify an upper limit. Specifying zero removes the upper limit.

Preventing non-owner file deletion in shared directories

You can protect files in shared directories from deletion by users who do not own them by setting the directory sticky bit with the **chmod** command. The sticky bit is one of a file's mode bits that controls permissions and other attributes. It was originally used to make an executable file "stick" in memory after execution. However, for directories, setting the sticky bit has a different meaning.

When the sticky bit is set on a directory that offers write access, only the owner of a file in that directory is allowed to remove the file. Without the sticky bit set, other users who do not have ownership or access to the file can remove it if they have write access to the containing directory.

For example, to set the sticky bit on **/tmp**, become root and enter the following command:

```
chmod +t /tmp
```

You can verify the change by entering:

```
# ls -ld /tmp
drwxrwxrwt  7 root    root    3072 Apr 27 17:00 /tmp
```

The `t` at the end of the first output field indicates that the sticky bit is set for the specified directory.

By setting the sticky bit on a publicly accessible directory, you may make it more difficult for users to clean up files. Only a superuser will be able to remove files owned by others in the directory that are unnecessarily taking up space, such as temporary files or files left behind by programs that fail.

If you want to provide this sort of file protection to your users, we recommend that you set the sticky bit on the following shared directories:

- **/tmp**
- **/var/news**
- **/var/preserve**

- **/var/spool/uucppublic**
- **/var/tmp**

Controlling the size of files and directories

Files are created in directories not only by users, but also by many of the processes that run on the DG/UX system. Sometimes interim or temporary files created by a process are not deleted. In some cases, processes create and add messages and information to log files that, over time, grow in size.

Directories are contained in file systems. When a file system becomes 80% full, I/O performance begins to decline. When a file system becomes 90% full, operations requiring more space will fail (such as creating new files or directories or increasing the size of a file). Also at 90% full, only the superuser can perform operations that involve allocating more space in the file system; any non-superuser process that tries to create a file or write to a file in a 90%-full file system will fail with an error such as `no space left on device`.

Therefore it is important to periodically:

- Ask all users to look over the files they own and delete any files they don't need. If a file might be needed at a later date, the file can be backed up onto some other medium before it's deleted.
- Review the contents of the system temporary file directories and delete unnecessary files. Again, a file can be backed up if it might be needed later. For more information, see "Cleaning temporary file directories."
- Clean the log files that are created and added to by processes. For more information, see "Cleaning log files."

Finding large files

The operation File System–> File Information–> Find searches for files based on a variety of criteria. For example, you can search for files of a given size, or you can search for files modified during the last week.

Finding out the size of a directory and its subdirectories

To display the size of any directory (including any subdirectories), use the **du** command. The **du** command returns the size in 512-byte blocks.

Finding out the number of used and free blocks and inodes

The **sysadm** operation File System–> File Information–> Disk Use shows the number of blocks used and free and the number of inodes (file slots) used and free.

Finding out the space allocation of a file system

You can use the **df** command to display the space allocation information for a file system. Following is a sample **df** display.

```
# df -lt / /usr ↵
/     (/dev/dsk/root      ):     8199 blocks    4869 files
                        total:    40000 blocks    5760 files
/usr    (/dev/dsk/usr     ):     1415 blocks   25895 files
                        total:   240000 blocks   34560 files
```

The display shows the mount point directory name, the pathname of the virtual disk device, the number of free (unallocated) 512-byte data blocks and file slots, and the total number of data blocks and file slots.

To expand the root (/) or **/usr** file systems, remember that you cannot boot from a file system built on a virtual disk that spans multiple physical disks. You need to boot from the root file system because it contains your kernel, **/dgux**. You need to boot from the the **/usr** file system because it contains stand-alone **sysadm**, **/usr/stand/sysadm**, which you may need to boot to recover after a failure.

To increase the size of either of these file systems, ensure that the physical disk housing the / or **/usr** file system contains sufficient space for the creation of another partition. The expansion generally results in an aggregated virtual disk. You may perform this operation while the / and **/usr** file systems are mounted and the system is online and in use.

For how to increase or change the size of a DG/UX file system, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Cleaning temporary file directories

In the / file system, there are several directories and files that you should check regularly to make sure they are not growing excessively. What constitutes excessive growth depends on how much free space you need in your file systems. The following sections describe these files and directories in more detail.

First among these directories are the system temporary file directories located in the / file system, **/tmp** and **/var/tmp**. Various programs use these directories to store temporary work files. Some applications normally do not remove their temporary files, and many applications leave temporary files behind if they terminate abnormally.

The easiest way to keep temporary file directories clean is by running a janitor-type job regularly using the **cron** utility. The prototype **crontab** file for **root** includes jobs that clean out **/tmp** and **/var/tmp** periodically. For more information on the **cron** utility and the prototype **root** jobs, see Chapter 6.

If you need to add to or modify the contents of a prototype file, make a copy of the file, using the same name but without the **.proto** suffix, and modify the file copy. Upgrades of the DG/UX system overload prototype files (files with the **.proto** suffix), but not the file copies without the suffix. Therefore modifying a copy of a prototype file preserves your changes during a DG/UX system upgrade.

Cleaning log files

The tables in this section list each DG/UX log file, its purpose, whether the controlling program appends or truncates it, and a method for cleanup.

The log files with "none" in their "Cleanup" columns take care of their own cleanup. If the "Cleanup" column in a table recommends special steps, see the section immediately following the table for the appropriate cleanup. If the column entry is "truncate," use this command:

```
# cat /dev/null > filename
```

Instead of truncating the file completely, however, to preserve the newer entries in a log file, deleting the older entries, you can use this command:

```
# tail filename > /tmp/filename; cat /tmp/filename >
filename
```

Cleaning up /etc

Table 4–7 lists the log files you need to clean up in the **/etc** directory.

Table 4–7 Log files in the /etc directory

Log file location	Log's purpose	Usage	Cleanup
/etc/log/fsck.log	fsck	truncated	truncate
/etc/log/fast_fsck.log	fast recovery fsck	truncated	none
/etc/log/filesave.log	filesave	appended	truncate
/etc/log/init.log	init	truncated	none
/etc/log/netinit.log	netinit	truncated	truncate
/etc/log/nfsfs.log	nfsfs mount requests	truncated	truncate
/etc/log/preserve.rclock	preserve start time	truncated	truncate
/etc/lp/logs/lpNet	lpNet start	appended	truncate
/etc/lp/logs/lpsched	lpsched start/stop	appended	truncate
/etc/lp/logs/requests	lp request	appended	truncate
/etc/wtmp	user data	appended	See section on truncating /etc/wtmp file
/etc/utmp	user data	truncated	none

Truncating the /etc/wtmp file

You should reduce the size of the **/etc/wtmp** file occasionally. The system logs accounting and user login information to this file on a continual basis, causing the file to grow. When the file becomes too large, you may choose either to remove it, replacing it with an empty **wtmp** file, or to reduce it, leaving only some of the more recent entries.

If you use accounting on your system, you should note that removing or reducing this file removes information required by the accounting system to charge for connect time. For more information on the accounting system, see Chapter 16.

If you remove the oversized **wtmp** file, remember to replace it with an empty file. If you choose to reduce the size of the file, you should note that this file is a data file (not a text file) made up of 64-character entries, and you should not edit it with a text editor. Instead, use the **tail** command to extract entries from the end of the file, the goal being to replace the **wtmp** file with these final entries.

On the **tail** command line, be careful to specify a number of characters that is divisible by 64, the size of a file entry. If any entry in the new **wtmp** file is incomplete, some commands may fail. For example, to reduce **wtmp**, leaving only the last 3200 characters, issue these command lines:

```
# tail  -3200c  /etc/wtmp  >  /tmp/wtmp ↵
# mv  /tmp/wtmp  /etc/wtmp ↵
```

For more information about the **wtmp** file, see the **wtmp** manual page.

Cleaning up /var

Table 4–8 lists the log files you need to clean up in the **/var** directory.

Table 4–8 Log files in the /var directory

Log file location	Purpose	Usage	Cleanup
/var/cron/log	cron start	appended	truncate
/var/lp/logs/lpNet	lpNet	appended	truncate
/var/lp/logs/lpsched	lpsched	appended	truncate
/var/lp/logs/requests	lp	appended	see section on lp print service logs
/var/setup.d/log/dgux.root	setup of root	appended	truncate
/var/setup.d/log/dgux.usr	setup of usr	appended	truncate
/var/setup.d/log/nfs.root	setup nfs root	appended	truncate
/var/set.d/log/onc.root	setup onc root	appended	truncate
/var/setup.d/log/tcpip.root	setup tcpip root	appended	truncate
/var/setup.d/log/tcpip.usr	setup tcpip usr	appended	truncate

Cleaning up LP print service logs

The LP print service has several logs that require occasional trimming. These logs, located in **/var/lp/logs**, are **lpNet**, **lpsched**, and **requests**. The LP print service's prototype **crontab** file, **/admin/crontabs/lp.proto**, contains jobs to prevent these logs from growing without limit. To use them on your system, execute **crontab –e** to edit the superuser's **crontab** file. Then add the jobs from the **lp.proto** file. For information on **cron**, see Chapter 6. For

information on the LP print service logs and the **lp.proto cron** jobs, see *Installing and Managing Printers on the DG/UX™ System*.

Cleaning up /var/adm

The **/var/adm** directory contains logs of various kinds that you should check occasionally. For example, if you use the system activity monitor, **sar** or **nsar**, you need to make sure that **sar** output logs in **/var/adm/sa** do not take up too much space.

If you use the accounting system, you need to check the **/var/adm** and **/var/adm/acct** directories occasionally to make sure they are not growing out of bounds. For information on these files and the accounting system, see Chapter 16.

If users on your system use the **spell** utility, you should occasionally check **/var/adm/spellhist** to make sure it is not growing excessively. The **spellhist** file contains words not recognized by **spell**. If your users have no use for the **spellhist** file, you may simply delete it; however, if they like to track words that appear there, you can at least reduce the size of the file by using **sort** to sort the file and remove duplicate entries. The following example demonstrates:

```
# cd /var/adm ↵
# sort -u spellhist > /tmp/spell.tmp ↵
# mv /tmp/spell.tmp spellhist ↵
```

Table 4–9 Log files in the /var/adm directory

Log file location	Log's purpose	Usage	Cleanup
/var/adm/acct/nite/fd2log	accounting	truncated	none
/var/adm/acct/nite/wmtp.*MMDD*	accounting data	truncated	none
/var/adm/acct/nite/wmtperror*MMDD*	accounting errors	truncated	none
/var/adm/log/backup.log	sysadm dump2	appended	truncate
/var/adm/log/idc.log	Idc compiler	appended	truncate
/var/adm/log/sysadm.log	sysadm	appended	truncate
/var/adm/messages	system messages maintained by syslog	appended	truncate
/var/adm/sa/*	sar data	See **sar(1)** manual page	

Continued

Log file location	Log's purpose	Usage	Cleanup
/var/adm/spellhist	data	appended	truncate
/var/adm/sulog	switch user	appended	truncate

Cleaning up /var/saf

The log files in **/var/saf** support **ttymon** and **sac.** When you clean up log files related to **ttymon**, you must be careful not to remove or recreate these files while either **ttymon** or **sac** is running. The **sac** log file is **/var/saf/_log**, and **ttymon** log files are stored in **/var/saf/***pmtag***/log**, where *pmtag* is the name of the port monitor.

If you are not interesting in saving the data from these log files and decide to truncate, use these commands:

For **sac**:

```
# cd /var/saf/;>log
```

For **ttymon**:

```
# cd /var/saf/pmtag;>log
```

If you want to preserve some of the latest data in the log files, use these commands instead:

For **sac**:

```
# tail /var/saf/_log >/tmp/saflog; cat /tmp/saflog >/var/saf/_log
```

For **ttymon**:

```
# tail /var/saf/pmtag/log >/tmp/pmtaglog; cat/tmp/pmtaglog \
>/var/saf/pmtag/log
```

The **>** character is a redirection operator.

Table 4–10 Log files in the /var/saf directory

Log file location	Purpose	Usage	Cleanup
/var/saf/tcp/log	tcp listen port monitor	appended	truncate
/var/saf/*pmtag*/log, where *pmtag* is the name of a port monitor	ttymon port monitor service	appended	truncate
/var/saf/_log	output from saf process	appended	truncate

Cleaning up /var/spool

Occasionally, you should check the **/var/spool** directory, which contains directories supporting a number of system services. For example, this directory contains the files and directories supporting the printer (LP) services. An interrupted LP command could abandon a file in the LP requests directory, for example. If such an accident happens often enough, you could waste valuable disk space. When you find lost print jobs, you can delete them or save them for the user who submitted the print job.

Table 4–11 lists log files in **/var/spool** that you should clean occasionally.

Table 4–11 Log files in the /var/spool directory

Log file location	Purpose	Usage	Cleanup
/var/spool/uucp/.Log/uucico/system	uucico	appended	truncate
/var/spool/uucp/.Log/uuxqt/system	uuxqt	appended	truncate

Cleaning mail system files

By default, the DG/UX system uses **/var/mail** as the mail directory for use by the **mailx** program. It is good practice to check **/var/mail** occasionally to look for:

- **Oversized mail files**

Simply enough, some users never delete mail. If you find that your **/var** directory is getting too full, check the **mail** directory to see if there are any excessively large files in **/var/mail**.

- **Misaddressed mail**

  Typing errors made by users sending mail can result in mail messages that sit indefinitely in files that no one reads. Depending on how your site is configured, your mail system will probably return mail to users if they accidentally send it to a nonexistent user. However, you may want to inspect the listing of mail files in **/var/mail** occasionally to make sure that only valid files, named for real users, exist.

  For example, some sites require users to use the **mailx** command instead of the **mail** command because the **mail** command cannot resolve network mailing addresses the way **mailx** can. Accidentally omitting the the **x** from **mailx** may thus result in a mail message being delivered to the local mail directory rather than to a remote system where it belongs.

Managing swap areas

A swap area is memory on a disk set aside for the system to use as paging space. Paging space can be thought of as short-term memory, a place to store data temporarily while other activities are going on.

Although a swap area is not a file system, you and the DG/UX system manage swap areas in much the same manner as file systems. For each swap area on your system, there is an entry in your file system table, **/etc/fstab**. Depending on how you configured your system, the system may check **fstab** at boot for any swap entries and use the listed virtual disks as swap areas. The Swap Area menu provides operations for managing swap area entries in **fstab**.

If you wish to use two physical disk areas for swapping, you should create a second virtual disk and add it as a separate swap area rather than creating a two piece virtual disk and swap on that one virtual disk. This is because the swapping code in the kernel will take advantage of the two swap areas to optimize swapping operations. If the two areas are bundled as a single virtual disk, the swapping code does not recognize they are separate and might not spread the swapping load between them.

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

By default, your system has one swap area 24 megabytes (50,000 512-byte blocks) in size. Your swap area may be different if your system is an OS client or if you changed the default swap size

during installation. If you know that your applications will need additional swap area, you may add more.

You should consider adding more swap space to a system when the number of users or the application load on the system increases, or if processes terminate unexpectedly and messages like this appear on your system console:

```
From system:  Out of paging area space
```

There is no universally applicable formula for calculating how much swap area you need. You need to consider how much memory your applications require and how much physical memory your computer has. A good general guideline is to start with swap area equal to 1.5 times your physical memory. For example, a system with 16 megabytes of memory should have 24 megabytes (50,000 blocks) of swap area.

A system may have swap space on its own disks, or it may have swap space on another system accessible over the network. You may not have swap space on a local disk as well as on a remote host's disk. Typically, a system that has its own disks with its own OS software will also have its swap area on a local disk, while an OS client will have its swap area on the server's disk. If any OS client has its own disk, it may create its swap area there even though its OS software comes from an OS server on the network. For more information on the local swap/remote OS configuration, see Chapter 18.

IMPORTANT: You are discouraged from using a nonvolatile random access memory (NVRAM) device for swap space. A NVRAM device does not have enough memory to be a reasonable swap resource.

Adding a swap area

To add a local swap space in addition to the default swap area already established for your system, use the **sysadm** operation `Device-> Disk-> Virtual-> Create` to create a virtual disk of the desired size. For how to create a virtual disk, see *Managing Mass Storage Devices and DG/UX™ File Systems*. You should not create a file system on a virtual disk to be used for swap space.

After you have created the virtual disk, add the swap area to **fstab** and make it usable on the system by selecting the **sysadm** operation `File System-> Swap Area-> Add`. The Add operation lists the available virtual disks, the ones not already appearing in **fstab**, and lets you choose one. It then adds the **fstab** entry and calls **swapon** to make the new swap area active immediately.

You cannot make a swap area inactive while the system is running. Once you have made a swap area active, it remains active until you shut down the system.

To make a swap area active every time you boot the system, you not only need to add an **fstab** entry, but you also need to change your system parameters so that the **swapon** program, when executed by the system at boot time, will run with the **-a** argument. Make this change by selecting the **sysadm** operation `System-> Parameters-> Set`. In the query for `arguments to swapon`, specify **-a**. The **-a** option to **swapon** causes it to mount all swap areas listed in the **fstab** file at boot time.

Expanding a swap area

In general, we recommend that you not expand the virtual disk containing swap area to increase swap area. Instead, we recommend that you create a new virtual disk and add it as additional swap area following the instructions in "Adding swap area."

However, you can expand the virtual disk containing swap area if empty blocks follow it on the physical disk (as shown by a layout listing from the **sysadm** Device–> Disk–> Physical–> List operation). Use the **sysadm** Device–> Disk–> Virtual–> Expand operation to expand the virtual disk to occupy the empty blocks. You must then reboot the system to cause the additional blocks to be used for swapping.

For how to expand a virtual disk, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Deleting a swap area

To remove a swap area entry from the **fstab** file, select the **sysadm** operation `File System-> Swap Area-> Delete`. Deleting a swap area entry does not immediately stop the system from using the swap area. The system will continue to use the swap area until you reboot.

Displaying swap areas

To list the swap areas in your file system table, select the **sysadm** operation `File System-> Swap Area-> List`. A sample swap area display appears below:

```
Swap Areas
--------------------------
/dev/dsk/swap
/dev/dsk/swap2
```

The display shows the pathnames of virtual disks entered in the file system table as swap area.

End of Chapter

5 Users, user groups, and user licenses

This chapter tells you how to set up and manage the everyday environment in which your computer users work. Tasks include creating login accounts for users, assigning initial user passwords, creating and maintaining user groups, and managing your user license.

The ability of a system to recognize users, passwords, and user groups changes if the system is part of a network using the Network Information Service (NIS) facility (formerly called Yellow Pages or YP). For information on how the NIS facility affects a system and password recognition, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

IMPORTANT: The user license defines the approved number of users that can use the DG/UX system concurrently on an AViiON® server or workstation. The user license mechanism of the DG/UX system enforces the maximum number of concurrent users of the system. This number must be registered with Data General and can be increased only by contacting Data General for a user license upgrade. For information on user licenses and how to update them, see "User licenses."

Users and login accounts

A user must have a login account to log in to the system. A user login account determines some of the characteristics of a user's environment as well as some attributes of programs the user runs. Local login account information is stored in the **/etc/passwd** file. Each line in the file represents one user's login account.

The following list shows how the **sysadm** defaults for user login accounts are originally set:

User Id
: Set to highest unassigned valid number plus one, as long as it is less than 60000; otherwise, lowest unassigned number over 100.

Group
: Set to **general**.

Home directory
: Set to **/home/***login_name*.

User comment
: No default.

Shell program
Set to **/sbin/sh**.

You can change the defaults using the **sysadm** operation `User-> Login Account-> Defaults-> Set`.

The home directory

The home directory, sometimes called the parent directory, is the primary space you assign to a user for file storage. In the process of creating a user login account, you are required to define a home directory for the user.

You can put a user's home directory in any existing file system, but we recommend that you develop a systematic approach to grouping user home directories.

The default home directory provided with the DG/UX system, **/home/***login_name*, suggests one approach to creating home directories: In a file system called **/home**, create a home directory for each user with the same login name as you give the user. For example, if you add a user with the login name **smithjj**, you define the home directory for **smithjj** as **/home/smithjj**. With this basic approach, you have the convenience of knowing that all user home directories are in **/home**.

However, you can organize home directories using any strategy that fits your needs. For instance, putting all user home directories in **/home** may prove unwieldy to manage as the number of users increases. To head off possible problems, you may decide to organize home directories by departments or groups within your company or organization.

If you have or plan to use ONC/NFS in the future or, in general, if you plan to make your system part of a network, you should make sure that parent directories have unique names across systems within the network. One way to do this is by using the host name as the home directory name, so that user home directories have the general form:

```
/host_name/login_name
```

This practice makes it easy to identify the origin and ownership of file systems in your network.

Shells available to DG/UX users

The shell is the program that provides your command line interface to the DG/UX system. The default shell you define for a user login

account is activated every time a user logs in to the system and controls how the user interacts with the system at the command line.

The following shells can be defined as a user's default shell:

- **/sbin/sh** (Bourne shell)
- **/sbin/csh** (C shell)
- **/usr/bin/ksh** (Korn shell)
- **/bin/restsh** (restricted shell)

In addition to these shells provided with the DG/UX system, the default shell can be any executable local shell program.

To find out what the Bourne and C shells do and how to use them, see *Using the DG/UX™ System*. For a description of the restricted shell and how it controls user access to a system, see Chapter 17.

Global and local profiles

The profile is the key element in establishing an environment in which users can be the most productive.

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 message-of-the-day file, **/etc/motd**, upon login. See Chapter 2 for information on **motd**.
- Commands to print notification of new mail messages.

There are two types of profiles: global and local.

The global profile is an ASCII text file, **/etc/profile** for **sh** and **ksh** users or **/etc/login.csh** for **csh** users, that helps to set up the user's working environment. The global profile contains commands, shell procedures, and environment variable assignments. When a user logs in, the **login** process executes the global profile for the user's initial shell. Users may further customize their personal environment by creating local profiles.

To see the global profile shipped for **sh** and **ksh** users, look at the **/etc/profile** file on-line. To see the global profile shipped for **csh** users, look at the the **/etc/login.csh** file on-line.

The local or individual profile is **.profile** for **sh** users, and **.login** for **csh** users. These files are stored in a user's home directory. The

local profiles are copies of two prototype files: **/etc/skel/.profile** and **/etc/skel/.cshrc**.

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** when a process begins. Since **login** is a process, the array of environment strings is made available to it. The local profiles in Figure 5–1 show how environment variables can be defined for users running the Bourne shell (**sh**) or the C shell (**csh**).

```
#LOCAL .PROFILE (sh or ksh)    #LOCAL .LOGIN (csh)
#                              #
#                              #
LOGNAME=poulet                 set prompt = 'sys5>'
PATH=:/bin:/usr/bin            set noclobber
MAIL=/usr/mail/poulet          setenv PATH $HOME/bin:/usr/bin:/bin
```

Figure 5–1 Environment variables in a local .profile file

Other programs make use of the information in the environment array list. New strings can be defined at any time. For a discussion of shell syntax, see *Using the DG/UX™ System* or the **sh** or **csh** manual pages.

User records in the password files

IMPORTANT: For how to change a password, see *Using the DG/UX™ System*.

Before users are permitted to log in to your system, they must be listed either in the local **/etc/passwd** file or, if your system is in an NIS domain, in the global NIS **yppasswd** 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,,,:/home/poulet:/usr/bin/csh
```

The fields are:

login name: poulet
: The login name (sometimes called a user name) is the name by which the system knows the user. A login name can be up to 32 characters long, must begin with an alphabetic character, and can contain alphanumeric characters (**a** through **z**, **A** through **Z**, **0** through **9**), dashes (-), underscores (_), and periods (.). Some existing DG/UX commands, other UNIX™ systems, and applications require login names no longer than eight characters to function correctly; if compatibility is important, we recommend limiting login names to eight characters. If you define login names longer than eight characters, we strongly recommend that the first eight characters be unique on your system or, if the system is part of an NIS domain, within the domain.

password: Rm27oQak1
: A user chooses a password and registers it with the system with the **passwd** 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, is an optional field that you can set to monitor password age and force users to change passwords periodically. For information on this mechanism and how to use it, see "Password aging.")

user ID: 103
: The user ID number (**uid**) is between 100 and 60,000. The number may not include any punctuation. Numbers 99 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. Can contain user's name, office, office phone number, home phone number, and so on; cannot contain a colon (:). Strictly speaking, there is no required format for this field. For historical reasons, this field is also called the **gecos** (JEE-cose) field. Some utilities, such as **finger**, expect the **gecos** field to be in a particular format. See the **finger** manual page.

home directory: /home/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 **/home**. 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 **/sbin/sh**. This field is most commonly used to invoke a special shell, such as **/bin/restsh** (restricted shell).

Password aging

Password aging is a security feature that lets you control how long a password can be in effect before the user must change it. If system security is very important to you, we recommend that you activate this feature. It forces users to change their passwords periodically and prevents them from changing a password before a specified time interval.

You can activate password aging for a login account when you execute `User-> Login Account-> Add`.

The password aging information is appended to the encrypted password field in the **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 **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 1062.

Table 5–1 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 5–1 Password aging 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 the superuser (**sysadm** or **root**) is able to change the password for that login.

Sample password file records with password aging set

Password administration can be set up in a variety of ways to meet the needs of different organizations. The following example 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 account entry in the **/etc/passwd** file for the typical user **bas**:

```
bas:mst3kmMOE2m.E:100:1:Bo Smith,,,:/home/bas:/usr/bin/csh
```

To require **bas** 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 **passwd** file, adding **,0/** to the password field, the entry looks like this:

```
bas:mst3kmMOE2m.E,0/:100:1:Bo Smith,,,:/home/bas:/usr/bin/csh
```

After the password entry is changed, **bas** will have to change the password at the next login and every 2 weeks thereafter.

After **bas**'s first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field.

```
bas:mst3kmMOE2m.E,0/W9:100:1:Bo Smith,,,:/home/bas:/usr/bin/csh
```

In this example, **bas** changed the password in week **W9**.

Adding a user login account

Use the **sysadm** operation User–> Login Account–> Add to add a new user's login account to your system. The operation adds an entry for the user to the appropriate password file. If the system is an NIS client, the local **/etc/password** file is used for the Add operation. If you set up the system as an NIS master, the global **yppasswd** database is used instead of the local password database. Refer to *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for information on NIS.

For each login account you add, you need to provide the following information:

Login Name

The login name is the name by which the system knows the user. A login name can be up to 32 characters long, must begin with an alphabetic character, and can contain alphanumeric characters (**a** through **z**, **A** through **Z**, **0** through **9**), dashes (-), underscores (_), and periods (.). Some existing DG/UX commands, other UNIX systems, and applications require login names no longer than eight characters to function correctly; if compatibility is important, we recommend limiting login names to eight characters. If you define login names longer than eight characters, we strongly recommend that the first eight characters be unique on your system or, if the system is part of an NIS domain, within the domain.

One strategy for creating login names is to use some form of the user's given name. For example, for a user named Martin Luther King, you could create a login name of **king** or **mlking**.

Add password aging for the login account [yes/no]

Answering **no** means you do not want to define an aging period. Answering **yes** means you want to define an aging period; you will have to provide two additional pieces of

information: **Minimum number of weeks before** *user* **may change his password** and **Maximum number of weeks until** *user* **must change his password**.

Minimum number of weeks before *user* **may change his password**

This query appears only if you selected password aging for the login account. Enter the minimum number of weeks that may pass before the user may change his password — a number in the range 0 to 64. The default, 0, lets the user change password immediately at login, which is generally desirable. If the user attempts to change his password before this time has passed, the attempt will fail. To let the user change his password at any time up to the expiration date, enter 0 (the expiration date is the maximum number of weeks before the user can change the password).

If both the minimum number of weeks and the maximum number of weeks are equal to 0, the user must change his password the first time he logs into the system. Setting the minimum and maximum both to 0 forces only one change of password for the account. After that change has occurred (even if performed by the superuser), password aging does not apply and the password may remain indefinitely.

If the minimum number of weeks is greater than the maximum number of weeks, only the superuser can change the password for this account.

Maximum number of weeks until *user* **must change his password**

This query appears only if you selected password aging for the login account. Enter the maximum number of weeks that may pass before the user must change a password — a number in the range 0 to 64. The default maximum aging period is 4 weeks. If the user does not change his password by this expiration period, the user will be prompted for a new password at the next login.

If both the minimum number of weeks and the maximum number of weeks are equal to 0, the user must change the password the first time he logs into the system. Setting the minimum and maximum both to 0 forces only one change of password for the account. After that change has occurred (even if performed by the superuser), password aging does not apply and the password may remain indefinitely.

If the minimum number of weeks is greater than the maximum number of weeks, only the superuser can change the password for this account.

User Id

The user ID, or UID, is a number between 0 and 60000. Internally, the system uses the user ID, not the login name, to identify a user. Numbers below 100 are reserved for the system accounts, so you should not assign user IDs in that range. Each UID on your system must be unique. If your system is in an NIS domain, the UID must be unique within the domain.

Other than serving as a unique identifier, the UID is arbitrary. As a default, **sysadm** supplies the value of the highest unassigned UID plus one, as long as the number is 60000 or less; otherwise, **the sysadm** default is the lowest unassigned UID.

Since the UID can serve as a unique identifier, you may decide to assign specific ranges of UID numbers to types of users as another means of identifying user subgroups.

Group

This is the name of an existing user group. Membership of users in a group grants them certain file and directory privileges while excluding other users' access. As a default, **sysadm** supplies the group name **general**.

Home Directory

This is the directory that will be the user's home directory and current directory upon logging into the system. As a default, **sysadm** displays the value of the base directory login account parameter (the shipped default is **/home**) plus the user's login name (for user **mlking**, the displayed home directory would be **/home/mlking**). If you want to give the user a home directory other than the default, you must specify the full path from the root.

If the home directory does not exist, an additional **sysadm** prompt will ask if you want to create it (**Create home directory for** *username*). If the home directory doesn't exist and you don't create it, the user will login to the root directory. We strongly recommend that you create separate home directories for users, to avoid space and clutter problems in the root directory and to provide each user with a private workspace.

User Comment

This comment may be any text that you choose to enter to describe the user or the login account. For example, you might enter the user's full name, the date, and the user's telephone number.

Shell Program

Select a shell for the user. The choice depends on the

standards you choose to define for your organization and the tastes of the user. Shells supported on the DG/UX system are **/sbin/sh** (Bourne), **/usr/bin/csh** (C), **/usr/bin/ksh** (Korn), and **/bin/restsh** (restricted). For a comparison of some of the shells, see *Using the DG/UX™ System*.

Set an initial password for *username* **[yes/no]**
Select this option if you want to assign a password for the user. We recommend a **yes** answer to this prompt. If you answer **no**, the user will not be able to login until you set a password. If you answer **no**, to change the password for such an account after creating it with the Add operation, use the **passwd** command (if the account is in the local **passwd** database) or the **yppasswd** command (if the account is in the NIS **yppasswd** database).

If you answer **yes** and specify a password, the password becomes effective immediately; you will need to tell it to the user so he or she can login. If you also specified a nondefault answer for password aging, the system will ask the user to change the password at the first login. If you leave the password blank, the user can log in without a password and can later set a password with the **passwd** command.

After you answer these queries, the Add operation proceeds to add the **passwd** entry.

If the user's home directory already exists or if the Add operation creates it, the Add operation copies to it some initial personal configuration files from a *skeleton* directory, **/etc/skel**. This skeleton directory exists simply as a prototype of a home directory, including several basic configuration files such as **.login** and **.profile** that serve as starting points for the user to begin customizing the working environment.

You may wish to create your own skeleton directory based on the one shipped with the system. Do not add or change files in the system default skeleton directory because future revisions of the DG/UX system may overwrite it with new contents. To set a different default skeleton directory name, use the operation `User-> Login Account-> Defaults-> Set`.

Displaying login accounts

Select the List operation to display login accounts. The operation lists information from the local **/etc/passwd** file unless you are a in an NIS domain, in which case the operation lets you choose which database to list.

You can restrict the login accounts listed by specifying a user ID, group ID, and login name. The default for all three fields is **all**. The operation lists accounts that match all three fields.

A typical listing of all accounts in a local login database looks like this:

```
Username        Uid     Gid  Home Directory     Shell
--------        ---     ---  --------------     -----
root              0       1  /                  /sbin/sh
xdm               0       1  /
/usr/bin/X11/telxdm
sysadm            0       0  /admin             /sbin/sh
daemon            1       1  /                  /sbin/sh
bin               2       2  /bin
sys               3       3  /usr/src
adm               4       4  /usr/adm           /sbin/sh
uucp              5       5  /usr/spool/uucp    /usr/lib/uucp/uucico
nuucp             5       1  /usr/lib/uucp      /sbin/sh
lp                6       2  /usr/lib           /sbin/sh
mail              8       1  /usr/mail
/usr/bin/mail
sync             19       1  /                  /bin/sync
yp               37      37  /usr/etc/yp        /sbin/sh
nfs              38      38  /                  /sbin/sh
ftp             127     127  /var/ftp           /sbin/sh
nobody        65534   65534  /
ted           17301     100  /home/ted          /usr/bin/csh
+                 0       0
```

The columns correspond to various fields of the **passwd** file. The last entry, **+**, indicates that the system also recognizes login accounts in the NIS database. See "User records in the password files" for more information on **passwd** entry format.

Modifying a login account

To change a user's login account, select the Modify operation. The Modify operation presents the same queries that you see when you use the Add operation to create the user login account.

The Modify operation looks for the login account in your system's local **/etc/passwd** file unless your system is the NIS master server, in which case the operation lets you choose whether to change the local **passwd** file or the global NIS **yppasswd** file.

Changes to a local login account will be in effect when the operation completes, but changes to a global NIS login account may not be in effect until the following day. Changes to a login name and user

number take effect throughout the system as soon as the **passwd** file changes; however, other changes to the login account (such as a changed home directory) will not take effect until the user logs in again.

Deleting a login account

To remove a user login account from the password database, select the Delete operation. The operation removes the login entry from your **/etc/passwd** file unless your system is the master server in an NIS domain, in which case the operation lets you choose whether you want to remove the entry from the global NIS **yppasswd** database or from your system's local **passwd** database.

The Delete operation lets you choose whether to delete the user's home directory or not. The operation does not delete the user's mail file (**/var/mail/***username*).

If you remove a user's login account while the user is logged in, the user does not experience any interruption in service. If the user logs out, however, the user will not be able to log in again.

After you have deleted a login account, the system can no longer resolve references to the user ID number and get the login name. This means that commands such as **ls** will return the user ID number instead of the name, where applicable. If you assign a new login account with an old ID number, the new login name will appear associated with all files that had been associated with the old login name. The algorithm used to determine a default UID for a new login account attempts to avoid this sort of confusion.

Resetting a forgotten password

Users sometimes forget their passwords. If the passwords on your system are not under NIS control (i.e., if your system is not an NIS client), you can reset the user password this way:

1. Become superuser on the system that controls the user login account.
2. Start an editing session for the local password file by entering:

   ```
   # vipw
   ```

   The **vipw** command starts the **vi** editor (or another editor you've defined as your default editor) with the local password file, **/etc/password**, as the file for editing.
3. Find and delete the contents of the *password* field in the record for the user's login account. Look for the user's login name in the first field. The *password* field is the second field in each record, immediately following the colon (:) after the login name.

4. Save your changes and quit the editor.
5. Set an initial password for the user by logging in with the user ID and running the **passwd** command to create a password.
6. Give the user the new password and instructions to reset the password immediately to comply with your organization's system security guidelines.

   If the system is part of an NIS domain, the passwords are under the control of the NIS server system. In this case, you need to reset the user password in the global NIS **yppasswd** file on the NIS server instead of in the local system's password file. For the global password file to change and how to change it, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

Setting and listing login account defaults

The Add operation of the Login Account menu supplies default values for queries where appropriate. To list the current defaults, use the operation `User-> Login Account-> Defaults-> Get`. If these default values do not suit your needs, you can change them with the operation `User-> Login Account-> Defaults-> Set`.

To set default login account parameters, follow these steps:

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

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

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

   ```
   Group: [general]
   ```

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

   The group name is limited to 32 alphanumeric characters; the first one must be alphabetic. However, we suggest shorter group names: eight characters or fewer. Specify a group name that is unique within the NIS domain. For information on user groups, see "User groups." For example:

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

3. The base directory is the parent directory of all users' home directories. The directory will contain the user's home directory. By default, the base directory is **/home**, so a new user's home directory is **/home/***username*. The base directory must exist, and you must specify its full path from the root (you should have already defined the mount point directory for user's home directories in your general disk setup). For example:

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

4. The skeleton directory contains the prototypes of configuration files such as **.profile**, **.login**, and **.cshrc** for each user's home directory. The default directory, **/etc/skel**, is included with DG/UX. The files in the skeleton directory provide a foundation from which the user may begin to customize his or her own operating environment. Do not add or change files in the system default skeleton directory because future revisions of the DG/UX system may overwrite it with new contents. If you wish, you may create your own skeleton directory. For example:

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

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

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

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

   You have created a set of default account parameters.

User groups

Groups are a means of establishing another level of ownership of and access to files and directories. Including users in groups allows you to grant them certain file and directory privileges while excluding users outside of the group.

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 as a secondary identification. By manipulating the permissions field of a file, the owner (or someone with the effective user ID of the owner) can grant read, write, or execute privileges to other group members.

A user may belong to more than one group, but a user's shell process is associated with only one group at a time. To change one's current group affiliation, use the **newgrp** command.

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, **general**, 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**.

The default group **general** is 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.

The **sysadm** Group menu provides operations for adding, deleting, modifying, and listing user groups. The **chmod** and **chgrp** commands control group privileges in the file system.

Local user group information is stored in the **/etc/group** file. Each line in the file represents a user group and consists 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
```

group_name: prog
: The group name can be up to 32 characters, although you may want to limit it to 8 characters to remain compatible with other systems. The first character 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 100 to 60000. 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.

Adding a user group

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

The **sysadm** operation User–> Group–> Add adds the entry to the local **/etc/group** file unless your system is the master server in an NIS domain, in which case it lets you choose whether to add the entry to the local **group** database or to the global NIS database.

You must create a user group before trying to add login accounts for users who will be members of the group.

To create a user group, follow these steps.

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

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

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

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

2. The group name can contain up to 32 alphanumeric characters; the first must be a letter. To remain compatible with pre-existing systems, we suggest that you limit group names to eight characters. A group name must be unique on the system or, if the system is part of an NIS domain, unique within the NIS domain. For example:

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

3. The group ID (GID) is a number from 0 to 60000. Numbers less than 100 are reserved for system use; do not assign them to users. The GID you assign to a new group must not be assigned already to another group name and must be unique on your system or, if the system is part of an NIS domain, unique within the NIS domain. As a default, **sysadm** supplies the highest assigned GID plus one. For example:

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

4. As group members, you specify the login names of users who will be members of the group, separated by commas. You need not specify login names when you create the group; when you add user accounts later, you can specify group names. If you do specify user login names, they must already exist as entries in the local **passwd** file or in the global NIS database, whichever applies. For example:

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

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

   You have created a user group.

Displaying user groups

Select the List operation to display entries from the **/etc/group** file. The List operation gets its information from the local **group** file unless your system is in an NIS domain, in which case you may choose between the local **group** file and the global NIS database.

An example of a local groups listing follows.

```
Group                    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                  100
+                          0
```

The display shows the group name, the group ID number, and the membership list. The last entry, +, indicates that the system also recognizes group accounts in the NIS database. See "Adding a user group" for more information on **group** entry format.

Modifying a user group

Select the Modify operation to change an entry in the **/etc/group** file. The operation changes the local **group** file unless you are the

master server in an NIS domain, in which case the operation lets you choose whether to change the local **group** file or the global NIS database.

The operation first prompts you for the name of the group you wish to change. The group must already exist. Then the operation prompts you with:

Group ID
The default is the group's current GID. If you wish to change the number, enter another. The ID number may not already exist. A GID must be between 100 and 60,000, inclusive.

Group Member Modification
For changing the list of group members, this query offers four choices:

no change
Select this value if you do not wish to change the membership.

append
Select this value to add users to the group. At the next prompt, **Group Members**, specify the login names to append to the current membership list.

remove
Select this value to delete users from the group. At the next prompt, **Group Members**, specify the login names to remove from the current membership list.

replace all
Select this value if you wish to replace the entire list of members with a new list. At the next prompt, **Group Members**, specify the new list of login names.

At the next prompt, **New Group Name**, you may specify a new name for the group if you wish.

Changes become visible in the local **group** file immediately. If you change the global NIS database, changes may not be visible until the following day.

Deleting a user group

Select the Delete operation to remove a user group. The operation removes the group from the local **/etc/group** file unless your system is the master server in an NIS domain, in which case it lets

you choose whether you want to remove the group from the local **group** file or from the global NIS database.

After you delete a group, the system can no longer resolve references to the group ID number and produce the group name. This means that commands such as **ls** will list the group ID number instead of the name, where applicable. If you add another group and give it the ID number of an old group, any files associated with the old group will now appear associated with the new one.

User licenses

The user license defines the approved number of users that can run the DG/UX system concurrently on an AViiON server or workstation. This number must be registered with Data General and can be increased only by contacting Data General for a user license upgrade.

No users other than **root** and **sysadm** (and **proto** or your designated model user on Trusted DG/UX systems) can log in to a system if the number of logged-in users is equal to the maximum user count permitted by the user license. However, users already logged in to the system can start another login process on that system regardless of the current user count.

The users **root** and **sysadm** (and the **proto** or your designated model user on Trusted DG/UX systems) are always allowed to log in to a system. Each of these users, when logged in, is included in the user count.

If a new user other than **root**, **sysadm**, or **proto** attempts to log in when the user count is at its maximum, the system denies access with the following message:

```
ERROR:  Login denied due to license restrictions
TO FIX: Consult your system administrator
```

A log entry is made and the email message is sent to **root** if the user count exceeds its maximum.

Listing user license information

Use the **sysadm** operation `System-> License-> List` to view license information. You select one of the following options under List to specify what you want to see:

Upgrade Info

Lists the information you'll need before you contact Data General and who to contact to request a license upgrade. Make sure that you have all of the listed information before you contact Data General to request a user license upgrade. Much of the information is the same as you provide on a standard order form, such as your company name and an address for invoicing. The following item is directly related to the user license:

Current licensed user count

The number of users currently approved to run the DG/UX system concurrently on the server or workstation.

Current License

Shows the number of concurrent users approved for the current DG/UX system license and the number of current users. By default, the number of current users is updated every 15 minutes. Since this number is updated periodically and not on demand, it may differ from the number of current users displayed by the Current Users operation, which is calculated on request.

Current Users

Lists the users recognized by the user count daemon as currently running DG/UX and shows information about these users:

```
User      pid    uid     TTY      Hostname
jones     16322  1300    pts/5    joe
pressley  9248   952     pts/4    fireball
palmer    2850   400     tty04
knightp   1145   550     pts/2    liberty
root      20184  0       tty14
```

The number generated by this operation may differ from the number of current users displayed by the Current License operation. This number is calculated on request, while the current user number for the Current License operation is updated periodically and not on request.

Upgrading a user license

Use the **sysadm** operation `System-> License-> Upgrade` to upgrade a user license after Data General approves an increase in the user count. At the prompt, enter the license token given to you by Data General after approval of your upgrade.

IMPORTANT: The upgrade mechanism is case-sensitive. You must enter the license token exactly as it is given to you by Data General, correctly typing uppercase and lowercase letters, to update the user license successfully.

When the Upgrade operation finishes, the new user count for the user license is in effect immediately.

End of Chapter

6 Jobs, processes, and system activity

This chapter explains how to automate execution of system management jobs, monitor and alter process activity, and monitor system activity.

Automating job execution

You may find it helpful to be able to schedule jobs to run on a regular basis. For example, you may have a script that checks disk free space and file system security. With the **cron** facility, you can schedule this script to run once a week or every night, for instance. The DG/UX system also offers the **at** and **batch** facilities, which run jobs at low priority or at any time that you specify. The following sections elaborate.

For more information on topics covered in this section, see manual pages for **cron**(1M), **crontab**(1), **at**(1), **atq**(1), **atrm**(1), and **batch**(1).

Scheduling jobs to run periodically (cron)

If you have certain jobs that you want run on a regular basis, you will find that the **cron** utility is one of your handiest tools. With **cron** you can schedule a job to run every five minutes, twice a week, or once a year, for example. Many subsystems of the DG/UX system, such as the LP print service and the accounting subsystems, include **cron** jobs for tasks such as collecting data and maintaining logs. You and other users on the system may also submit your own **cron** jobs.

Setting up a **cron** job involves executing the **crontab** command to start an editing session for your **crontab** file. Your **crontab** file contains a one line entry for each scheduled **cron** job. An entry comprises two kinds of information: the frequency of the job and the command line to be run. When you finish editing the file and exit the editor, **crontab** submits your new **crontab** file to **cron**. At the appointed times, **cron** runs the jobs, mailing you the output if any. Note that before other users can create a **crontab** file, a superuser must add their user names to the **/etc/cron.d/cron.allow** file.

▶ To schedule system administration **cron** jobs as superuser:

1. **Issue this command:**

```
# crontab -e ↵
```

If there is no **crontab** file for the superuser on your system, the **crontab** command will create an empty one; otherwise, the command opens an editing session with the existing **crontab** file. The **crontab** command invokes the **ed**(1) editor unless your EDITOR environment variable specifies another.

2. **Using the editor, insert the desired entries. An entry has the following format:**

```
minute  hour  day-of-month  month  day-of-week
```

where:
minute is in the range 0 – 59.
hour is in the range 0 – 23.
day-of-month is in the range 1 – 31.
month is in the range 1 – 12.
day-of-week is in the range 0 – 6, with 0 = Sunday.

When specifying multiple values in a field, separate the values with a comma (,). In any field you may use the asterisk (*) to represent all possible values. The following sections provide examples of and suggestions for **cron** jobs.

IMPORTANT: If you change the time or time zone, or if either daylight savings time (DST) or standard time (ST) begins, stop and restart **cron.**

3. **Save the file and exit from the editor. The new cron jobs will run at the appointed times.**

IMPORTANT: The **crontab** command will not submit the jobs if the editor returns an exit code other than 0. See the man page or other documentation for your editor. Some editors, such as the DG/UX editors **ed** and **vi**, return a non-zero exit code if you remove all lines from the file. In this case, **crontab** does not submit the file to **cron**, and your previous table of **cron** jobs remains in effect. The proper way to remove all **cron** jobs is to invoke **crontab** with the **-r** option.

You know that **crontab** has submitted your edited file of jobs when it returns this message:

```
warning: commands will be executed using /usr/bin/sh
```

This warning does not indicate a problem; rather, it serves to remind you that the command lines in your job entries should adhere to Bourne shell syntax. See **sh**(1) for more information on the Bourne shell.

Here are some suggestions for scheduling **cron** jobs:

- Try to schedule jobs for off-peak hours, particularly if the job is expensive in terms of resources like CPU time or disk access. Running during off-peak hours, the job is less likely to inconvenience other users. Example 2 demonstrates this point.
- Schedule jobs to run at odd minutes or hours to avoid coinciding with other jobs. If you and other users typically run your jobs only on the hour or half hour, you may find a noticeable degradation in system performance at these times if multiple jobs run simultaneously. Example 3 demonstrates this point.
- Minimize security risk by specifying complete pathnames of commands and by executing only commands residing in secure directories. A **cron** job executes as the user who submitted the job; therefore, it is particularly important that jobs to be run as superuser execute only secure commands. Scripts especially should have permissions set to prevent tampering.

Examples of job scheduling

The following entry schedules a **wall**(1M) message to send out a reminder every Tuesday morning at 9:00:

```
0 9 * * 2 /bin/echo /bin/echo "Meeting at 9:30!" | /etc/wall
```

The following entry schedules a file system security checking job to run on Monday, Wednesday, and Friday mornings at 2:15:

```
15 2 * * 1,3,5 /usr/sbin/admfsinfo -lq -o check
```

The **cron** facility will use **mail** to send you the output from the command line.

The following entry schedules a script to run every hour, Monday through Friday, at 10-minute intervals starting at 3 minutes past the hour. The script appends output to a file:

```
3,13,23,33,43,53 * * * 1-5 /admin/getCPUload >> /admin/load.log
```

The following entry schedules a message to run at 2:07 in the afternoon on June 20:

```
7 14 20 6 * /bin/echo /bin/echo "Happy Summer Solstice!" | /etc/wall
```

The following entry schedules the **ntpdate** function to poll a clock server on your LAN to reset your host time every eighth minute of each hour.

```
8 * * * * /usr/bin/ntpdate commtg3 brewery dg-rtp >/dev/null 2>&1
```

Jobs you can schedule to automate system administration tasks

Although the DG/UX system does not by default have any scheduled **cron** jobs, it does provide files containing prototype jobs that you may adapt or use as shipped. The files are in the **/admin/crontabs** directory.

Accounting and file cleanup

The **/admin/crontabs/root.proto** file contains two kinds of jobs: those helpful on systems running accounting, and those helpful for systems in general. The file contains:

```
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 -mount -atime +3 \
  -type f ! -name '[XM][0-9]*' -exec rm {} \;
15 3 * * 6 /bin/find /var/adm/log -name \
  'sysadm.log.[0-9][0-9][0-9]' -atime +7 -exec rm {} \;
50 9,3 * * * /bin/find /var/spool/cron/atjobs
  -name '.nfs*' -atime +1 -exec rm {} \;
5 4 * * 6 /usr/lib/newsyslog >/dev/null 2>&1
```

Some lines have been broken for readability. The first four jobs perform accounting functions. If your system does not use accounting (covered in Chapter 16), you do not need to schedule these jobs. The fifth through the eighth jobs clean up temporary file directories and remove outdated logs and some other files.

Printer maintenance

The **/admin/crontabs/lp.proto** file contains jobs to help maintain the LP system. You should schedule these jobs on any system that uses printers. The file contains these jobs:

```
13 3 * * * cd /var/lp/logs; if [ -f requests ]; then \
  /bin/mv requests xyzzy; /bin/cp xyzzy requests; \
  >xyzzy; /usr/lbin/agefile -c2 requests; /bin/mv \
  xyzzy requests; fi
15 3 * * 0 /usr/lbin/agefile -c4 /var/lp/logs/lpsched
17 3 * * 0 /usr/lbin/agefile -c4 /var/lp/logs/lpNet
```

The job beginning on the first line has been broken over multiple lines for readability. You should run these jobs as **lp**. To schedule these jobs on your system, first give **lp** permission to run **cron** jobs by adding an **lp** entry to **/etc/cron.d/cron.allow**. Then execute **su lp** to become **lp** before executing **crontab –e** to schedule the desired jobs. For information on the LP print service, see *Installing and Managing Printers on the DG/UX™ System*.

UUCP maintenance

The **/admin/crontabs/uucp.proto** file contains jobs for maintaining the UUCP file transfer and remote command execution facility. The prototype jobs are:

```
39,9 *   * * *  /etc/uucp/uudemon.hour  > /dev/null
10  *    * * *  /etc/uucp/uudemon.poll  > /dev/null
45  23   * * *  /etc/uucp/uudemon.cleanup > /dev/null
48  10,14 * * 1-5 /etc/uucp/uudemon.admin  > /dev/null
```

If you use UUCP, these jobs should run as **nuucp**, the login name intended for UUCP administration. To schedule these jobs on your system, first give **nuucp** permission to run **cron** jobs by adding an **nuucp** entry to **/etc/cron.d/cron.allow**. Then execute **su nuucp** to become **nuucp** before executing **crontab –e** to schedule the desired jobs. See *Using Modems and UUCP on the DG/UX™ System* for more information on UUCP.

Maintaining the cron log

The **cron** utility logs a history of its activity to the **/var/cron/log** file. You should periodically truncate the **/var/cron/log** file to prevent it from using too much disk space.

▶ To remove remove lines from the **cron** log file:

1. **Stop cron by executing the following command line:**

```
# /usr/sbin/init.d/rc.cron stop ↵
```

2. **Use the following command line to remove all but the last 100 lines of the log file:**

```
# tail -100 /var/cron/log > /tmp/log; mv /tmp/log /var/cron/log ↵
```

3. **Restart cron with the following command line:**

```
# /usr/sbin/init.d/rc.cron start ↵
```

Submitting jobs for delayed execution (at)

To run a job at a specified time on a specified date, use the **at**(1) command. This command is useful for running jobs during off hours or at any time when you may not be available or may forget to run the job yourself.

For example, to run the script **/admin/check_disks** at 3:00 in the morning on January 24, issue this command line at the shell prompt:

```
# echo /admin/check_disks | at 3:00 january 24 ↵
```

To broadcast a reminder about a meeting one hour from now, issue this command line:

```
# echo "echo Meeting at 2pm | wall" | at now + 1 hour ↵
```

To reboot the system at 11:00 tomorrow night, use this command line:

```
# echo init 6 | at 23:00 tomorrow ↵
```

IMPORTANT: The method of rebooting shown above is not recommended for active systems. For information on booting the system, see Chapter 3.

That **at** command accepts date and time specifications in a variety of formats. For more information, see the **at**(1) manual page. The **at** command submits jobs to the **a** queue managed by **cron**. For more information on queues, see the **cron**(1M) manual page.

Submitting low-priority jobs (batch)

There are times when you want to execute a job immediately but hesitate to do so because CPU time is in demand. At these times, the **batch** command is useful because it submits the job to a low-priority queue intended to minimize impact on system performance. As with the **at** and **cron** utilities, the **batch** utility mails you any output from the job.

For example, to submit the script **/admin/check_disks** at a low priority, use the following command line:

```
# echo  /admin/check_disks  |  batch ↵
```

The **batch** command submits the job to the **b** queue managed by **cron**. For more information, see the **batch**(1) and **cron**(1M) manual pages.

Shifting workload to off-peak hours

Use the **crontab** command to examine users' **crontab** files to see if there are jobs scheduled for peak hours that could just as well run during off hours. You may find the accounting system (Chapter 16) and the Process menu's List operation (which invokes the **ps**(1) command) helpful in determining what processes have the greatest impact on system performance.

Encourage users to run large jobs that are not interactive (such as program compilations) at off-peak hours. You may also want to run such jobs with a low priority by using the **nice**(1) or **batch**(1) commands. As superuser, you can always change a job's priority with the **renice**(1M) command.

Monitoring process activity

Select the **sysadm** `System-> Process` menu to monitor and control processes running on the system. A process is any program currently running on the system. The term *process* refers not only to the executing program code but also to the program's environment and state for that instance of execution.

The Process menu provides operations for deleting processes, changing process priority, listing processes, and sending signals to processes.

Deleting processes

Select sysadm operation `System-> Process-> Delete` to terminate a process. Deleting a process removes it completely from memory and from the operating system tables. Deleting a process may also delete any of the process's child processes.

The Delete operation restricts the field of processes that you may delete according to criteria that you specify:

`Process ID`
: Enter the ID numbers of existing processes.

`Owner login name`
: Enter the login names of users whose processes you wish to delete.

`Terminal ports`
: Enter terminals (TTYs) whose associated processes you wish to delete; for example, **console**, **tty04**, **ttyp7**, and so on.

The operation then uses the **ps**(1) command to assemble a list of processes that satisfy the stated criteria. The operation prompts

with `Process(es) to Delete`, letting you select processes from a list.

After the Delete operation has derived the desired process IDs based on your selections, it attempts to remove the processes with the equivalent of this command line:

```
# kill process_ID ↵
```

If this command does not succeed in killing them, it uses the equivalent of this command line:

```
# kill -9 process_ID ↵
```

Modifying processes

Select the **sysadm** operation `System-> Process-> Modify` to change the priority of a running process. The priority of a process determines in part how much CPU time the process scheduler gives the process.

Every process has a priority value associated with it when it starts. In a possible range of 0 to 39, 0 is high priority and 39 is low priority. A process with a low priority number will tend to get more CPU time than a process of a higher priority number. The normal user process has a priority of 20.

You can alter processes' priority levels to reflect the importance of the jobs on your system. For example, if the daily backup process is competing with an urgent batch job, you can lower the priority (raise the priority number) of the backup process (the **dump2**(1M) command) to improve performance of the batch job.

The Modify operation restricts the field of processes that you may modify according to criteria that you specify:

`Process ID`
: Enter the ID numbers of existing processes.

`Owner login name`
: Enter the login names of users whose processes you wish to modify.

`Terminal ports`
: Enter terminals (TTYs) whose associated processes you wish to modify; for example, **console**, **tty04**, **ttyp7**, and so on.

The operation then uses the **ps**(1) command to assemble a list of processes that satisfy the stated criteria. The operation prompts

with `Process(es) to Modify`, letting you select processes from a list.

You then select a new priority in the range 0 to 39, and the operation changes the process priorities with an equivalent of the **renice**(1M) command.

Displaying processes

Select the **sysadm** operation `System-> Process-> List` to display a list of processes currently executing on the system. The List operation calls the **ps**(1) command to get the process status information. The operation lets you restrict the report to selected processes, accepting three kinds of selection criteria:

`Process ID`
: Enter the ID numbers of existing processes.

`Owner login name`
: Enter the login names of users whose processes you wish to list.

`Terminal ports`
: Enter the names of terminals with which processes are associated, for example, **console**, **tty04**, **ttyp7**, and so on.

You may select whether to display a long listing or the default listing. The default listing corresponds to the **ps** command's **-f** (full) listing. The long listing corresponds to the **ps** command's **-l** (long) listing. See the **ps**(1) manual page for more information.

Getting process information (moved here from elsewhere in M4)

To obtain information about active processes, use the **sysadm** operation `System-> Process-> List`. This operation calls the **ps**(1) command to produce a listing.

The listing constitutes 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 process that uses progressively more system resources over a period of time while you are monitoring it, you should probably stop the process with the operation `System-> Process-> Delete`.

If you regularly run processes that take a very long time to execute, you should consider using **cron**(1M) or **at**(1) to execute the job

during off-hours, or use **batch**(1) to execute the job when system load level permits. See "Automating job execution" for information on these commands.

Signaling processes

Select the **sysadm** operation `System-> Process-> Signal` to send a signal to one or more processes. The effect of the signal depends on the receiving process. To kill a process, use `System-> Process-> Delete`, or use the Signal operation to send signal 15. If signal 15 does not kill the process, use signal 9. For more information on signals, see the manual page for the **kill**(2) system call.

The Signal operation restricts the field of processes for the operation based on criteria that you specify:

`Process ID`
: Enter the ID numbers of existing processes.

`Owner login name`
: Enter the login names of users whose processes you wish to signal.

`Terminal ports`
: Enter terminals (TTYs) whose associated processes you wish to signal; for example, **console**, **tty04**, **ttyp7**, and so on.

The operation then uses the **ps**(1) command to assemble a list of processes that satisfy the stated criteria. The operation prompts with `Process(es) to Signal`, letting you select processes from a list.

After you have selected or entered the desired signal, the Modify operation sends the signal to the processes.

Monitoring system activity

The DG/UX system activity reporting utilities **sar**(1) and **nsar**(1) let you gather and review statistics on CPU performance, disk and terminal I/O, memory usage, process communication and execution, paging and memory usage statistics, and other activity. When the system appears to be functioning erratically, or simply as a matter of course, you may want to review the system activity data.

For complete information on these utilities, see the **sar**(1), **sar**(1M), **nsar**(1), and **nsar**(1M) manual pages. For how to use **sar** and **nsar** to assist you in improving system performance, see *Analyzing DG/UX System Performance*.

The **sysadm** `System-> System Activity` menu provides operations for starting and stopping **sar** or **nsar**, deleting old data collections, and reviewing reports.

Starting system activity monitoring

Select the **sysadm** operation `System-> System Activity-> Start` to begin collecting data. The Start operation presents these prompts:

`Data Collection Name`
: Enter a file name to be used for the collection of data. If a data collection by the same name already exists, the monitor will append the new data to the existing file. If you do not specify a collection name, the operation will create a name of the form **spd.Day***nn*, where *nn* is the day of the month.

`Interval Between Samples (Seconds)`
: Enter the number of seconds that the system monitor should wait between taking samples. For best results, the interval should be not less than five seconds (the default) nor greater than a few minutes.

`Number of Samples`
: Enter the number of samples that you want the system monitor to take. Sampling ends when the number of samples specified is reached. Keep in mind that each sample is over 1,000 bytes in size.

The system monitor stores the data in the **/var/adm/sa/spd.***name* file. The file remains until you delete it explicitly with the shell's **rm**(1) command or with the `System-> Activity-> Delete` operation.

You may run multiple system monitoring sessions at the same time.

Stopping system activity monitoring

Select the **sysadm** operation `System-> System Activity-> Stop` to halt a data collection session before the monitor has collected the specified number of samples. You may choose any existing data collections for the Stop operation.

Stopping a monitoring session does not remove the associated data file from **/var/adm/sa**. You may list the data from a prematurely stopped monitoring session the same as for any monitoring session.

Deleting a system activity monitoring data set

Select the **sysadm** operation `System-> System Activity-> Delete` to remove one or more data collection files. Data collection

files, located in **/var/adm/sa**, take up more than 1,000 bytes per sample, so you should delete them when no longer needed.

Displaying system monitoring data

Select the **sysadm** operation `System-> System Activity-> List` to review the data collected during a system monitoring session, select the List operation. You may list data from a monitoring session that is still in progress or from a session that has completed. If you list data from a monitoring session that is still in progress, the display includes all data collected up to that time.

The List operation lets you choose the kind of information you want to see in the report. Each category corresponds to a particular option of the **sar**(1) command, the program that collects the data. The data categories (and corresponding **sar** options, in parentheses) are:

`All data (A)`
: Data for all available categories.

`File access (a)`
: Use of system routines used for file access.

`I/O buffer activity (b)`
: Activity in system buffers, including hit ratios for system caches.

`System calls (c)`
: Number of system calls served for all system calls and for some specific system calls.

`Disk usage (d)`
: Physical disk I/O activity. Disk names displayed are long device specifications. To see how device specifications map to their entries in the **/dev** directory on your system, see the file **/etc/devlinktab**.

`Interprocess communications (m)`
: Interprocess message (**msgsnd**(2)) and semaphore (**semop**(2)) activity.

`Run queue and paging (q)`
: Number of processes running and waiting to run.

`CPU utilization (u)`
: CPU usage by user and system processes and idle time.

`Kernel tables (v)`
: Number of entries in the process table, inode table, file table, and shared memory record table.

`Paging I/O and process switches (w)`
: Process swapping and switching activity.

`Paging rates (p)`
: Paging activity such as virtual page faults and physical page faults.

`Free space (r)`
: Unused memory pages and swap area (disk) blocks.

`Terminal I/O activity (y)`
: Terminal (TTY) I/O activity.

For more information on **sar** output, see **sar**(1) and **sar**(1M).

End of Chapter

7 Error logging, system dumps, and failure recovery

This chapter explains how to monitor system health, get information on system and network errors and problems, circumvent and recover from failures, and create a system dump and diagnostic tape to submit to Data General with a Software Trouble Report (STR).

Monitoring system health

A number of system facilities produce messages describing errors, abnormal conditions, routine checkpoints, and a number of other events. These facilities use the system error logger daemon, **syslogd**(8), to direct the messages to appropriate destinations, such as the system console, a user terminal, and files such as **/var/adm/messages**.

It is good practice to review **/var/adm/messages** occasionally to verify that your system is operating as expected. Some messages may indicate conditions requiring your attention. For how to generate log reports and change how the system handles these messages, see "Using the system error log," "Using the network error log," and the **syslog.conf**(5) manual page.

If you run the **/usr/sbin/newsyslog** script periodically, **/var/adm** may also contain old message files named **messages.0**, **messages.1**, **messages.2**, and **messages.3**. The **newsyslog** script is one of the jobs in the prototype **root crontab** file. If you install the jobs in this prototype file, **newsyslog** will run once a week, renaming the various **messages** files so that you can more easily see how old they are. For more information on the **cron** utility and the prototype **root** jobs, see Chapter 6.

Using the system error log

You establish the conditions under which system errors are collected, identify the logging repositories, and generate error log reports.

Turning on logging

Select `Logging-> System-> Add` to enable specific types of error logging. The Add operation presents you with these prompts:

`Facility Producing Error`

Enter the origin of the message being logged. Valid values for message originator are:

Table 7–1 Originators for system messages

Originator	Description
`user`	Messages generated by user processes. This is the default.
`kern`	Messages generated by the kernel.
`mail`	The mail system.
`daemon`	Daemon system servers such as **ftpd**.
`auth`	The authorization system: **login**, **su**, and **ttymon**.
`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**.
`news`	Reserved for the USENET network news system.
`uucp`	Reserved for the UUCP system.
all	Indicates all facilities except the mark facility.

`Error Severity Level`

Enter the severity level. Valid severity levels, in descending order of severity, are:

Table 7–2 Severity levels for system messages

Severity level	Description
`emergency`	Panic conditions that normally would be broadcast to all users.
`alert`	Conditions that should be corrected immediately, such as a corrupted system database.
`critical`	Messages about critical conditions, such as hard device errors.
`error`	Other errors.

Continued

Table 7–2 Severity levels for system messages

Severity level	Description
`warning`	Warning messages.
`notice`	Conditions that are not caused by an error, but may require attention.
`information`	Informational messages.
`debug`	Messages that normally are used when debugging a program.
`none`	No messages.

`Logging Destination:`

Enter where you want to forward the message to:

Table 7–3 Destinations for system messages

Destination	Description
save to file	The system prompts for the file name, which begins with a leading slash (/).
write message to user(s)	The system prompts for the username of the recipient (such as root) or a list of user names separated by commas. The message is written to the screens of all recipients in the list who are currently logged in.
send to remote host	The system prompts for the remote host name.

Deleting log selections

Through Delete, you can delete a logging selection by the facility producing the error, severity level, and logging destination for a particular host. Select `Logging-> System-> Delete`. The Delete operation presents you with this prompt:

`Entry to Delete:`

There is no default. Enter the selection by facility, level, and action to be deleted. Type `?` for a list, from which you can select entries for deletion.

Modifying log selections

Through Modify, you can change any of the settings previously made when you enabled logging. Select `Logging-> System-> Modify` to alter error logging. The Modify operation presents you with this prompt:

`Entry to Modify:`

Enter the system logging entry to be changed in the database. Type **?** to get a current list, an example of which follows:

```
Choices are

     1  *.err            /dev/console
     2  kern.debug       /dev/console
     3  auth.notice      /dev/console
     4  *.err            /usr/adm/messages
     5  kern.debug       /usr/adm/messages
     6  daemon.notice    /usr/adm/messages
     7  auth.notice      /usr/adm/messages
     8  mail.crit        /usr/adm/messages
     9  kern.crit        /var/adm/dgsvcmgr/log.com
    10  user.info        /dev/console
    11  *.alert          root
    12  daemon.info      /var/adm/daemon.info
    13  *.emerg          *
    14  daemon.notice    /var/adm/log/admd.log
    15  daemon.notice    /var/adm/messages

Enter a number, a name, the initial part of a name,
<NL> to take the default,? for help, ^ to return to
the previous query, < to restart the operation,or q
to quit.
```

Supply the number of the entry to modify. Remaining prompts are identical to those presented for the Add operation.

Listing system log selections

Through List, you can display the list of facilities and severity levels that are being logged. Select `Logging-> System-> List`. Sample output from this selection follows:

```
Facility    Level       Destination
------      -----       -----------------------
*           error       file /dev/syscon

kernel      debug       file /dev/syscon

auth        notice      file /dev/syscon

*           error       file /usr/adm/messages

kernel      debug       file /usr/adm/messages

daemon      notice      file /usr/adm/messages
auth        notice      file /usr/adm/messages

mail        critical    file /usr/adm/messages

kernel      critical    file /var/adm/dgsvcmgr/log.com

user        info        remote host jester
*           alert       write to root

daemon      info        file /var/adm/daemon.info

*           emergency   all logged-in users
```

An asterisk (*) selects all facilities.

Generating a log report

The Report operation generates a report of system errors that have been logged to files on the target host. Select `Logging-> System-> Report`. The Report operation presents you with these prompts:

`Originating Host(s):`

Enter the names of the hosts whose messages you want to see. Only local log files are consulted, but those files may include messages sent from other hosts by way of remote host entries. To specify a list, separate each host name with a comma. The default is all hosts.

`Identifier(s):`

Enter the names of the programs that report the system

error. To specify a list, separate each identifier name with a comma. The default is all identifiers.

`Message(s):`

Allows you to provide a regular expression to match the type of error messages you are interested in. You may need to view all messages so that you can derive a meaningful pattern. For example, to see only NetWorker-related errors, you can enter this regular expression:

```
Net.*
```

to match the pattern:

```
NetWorker savegroup: info patriot_full (with
one client)
```

Sample output follows:

```
Date             Host       Identifier   Message
---------------  ---------  -----------  ------------------
Feb 19 23:49:35  patriot    dg/ux        Tape device at
                                    st(cisc@28(FFFFF300,7),5,0
Feb 19 23:49:35  patriot    dg/ux         encountered a hard
                                        error at block
                                       0,status = 4005007
Feb 19 23:48:15  patriot    syslog        NetWorker media:
                                        (notice) 8mm tape
                        patriot_week3_a used 274 MB of 2000
Feb 19 23:48:10  patriot    dg/ux         Tape device at
                                    st(cisc@28(FFFFF300,7),5,0)
Feb 19 23:48:10  patriot    dg/ux         encountered a hard
                                        error at block
                                       0, status = 4005007
Feb 19 09:52:37  patriot    syslog        NetWorker index:
                                        (notice) cross-
                                       check has completed.
Feb 19 09:52:23  patriot    syslog        NetWorker index:
                                        (notice) nsrck has
                                       completed.
Feb 19 09:52:12  patriot    syslog        NetWorker Server:
                                        (notice) started
Feb 19 09:51:03  patriot    dg/ux         Firmware in SCSI
                                        controller
                          cisc@28(FFFFF300,7) is out of date
Feb 19 09:51:03  patriot    dg/ux         see release notice.
Feb 19 09:51:03  patriot    dgsvcd        AV/Alert System:
                                       Disabled
Feb 19 09:47:06  patriot    syslogd       going down on
                                       signal 15
Feb 19 09:47:05  patriot    dgsvcd        AV/Alert System:
                                       Going down on
                                      signal 15
```

Using the network error log

You establish the conditions under which network error messages are listed and deleted. The errors reported are dependent on the software that is installed. LAN device drivers, TCP/IP, X.25, or OSI can report such errors.

Select `Logging-> Network-> List` to view the network error logs. The List operation presents you with these prompts:

`Age (in days):`
: Enter the age (in days) of the oldest logged message to view. The default is three days. The default would present messages up to and including three days old, including one and two days old.

`Facility Producing Error:`
: Enter the origin of the message being logged. The list produced depends on the software that is installed. By default, messages from all sources are listed. Type `?` for a list of sources.

`Error Severity Level:`
: Enter the severity level. Recognized values, in descending order of severity, are:

Table 7–4 Severity levels for network messages

Severity level	Description
emergency	Panic conditions that normally would be broadcast to all users.
alert	Conditions that should be corrected immediately, such as a corrupted system database.
critical	Messages about critical conditions, such as hard device errors.
error	Other errors.
warning	Warning messages.
notice	Conditions that are not error conditions, but may require attention.

Continued

Table 7–4 Severity levels for network messages

Severity level	Description
`information`	Informational messages.
`debug`	Messages that normally are used when debugging a program.

Deleting log messages

Through Delete, you can delete log messages by age. Select `Logging-> Network-> Delete`. The Delete operation presents you with a single prompt:

`Age (in days):`
Enter the age (in days) of the logged messages you want to keep. The default is three days. All messages older than the specified age will be deleted.

Using the watchdog timer to detect and recover from system hangs

Introduced in DG/UX 5.4R2.10 to support the AV8500, AV9500 and AV/5500 systems, the watchdog timer performs an automatic system reset upon detection of either a hardware or software system hang. On AV8500 and AV9500 systems, a reset starts powerup diagnostics, and isolates and deconfigures any faulty field replaceable hardware components when the system is rebooted. If AV/Alert (a comprehensive diagnostic support system for AViiON family hardware) is enabled, the faulty hardware is also reported automatically to the Data General Customer Support Center.

DG/UX 5.4R3.00 extended support of the watchdog timer feature to cover the AV4500, AV450, and AV550 systems, as well as all AViiON systems that use the failover monitor **failovermon**(1M). The failover monitor **wdt** should be used in conjunction with DG/UX 5.4R2.10 hardware **wdt** when possible. For more information on the failover monitor, see *Achieving High Availability on AViiON® Systems*

Two panic codes may be produced during a hardware or software system hang: 53000060 or 53000061. Panic code 53000060 produces a hardware watchdog timer reset only. Since the current state of the job processors is not saved with a system reset, a dump of the memory image is not taken. Panic code 53000061 causes the generation of a memory image that can be dumped to tape or disk for analysis.

Enabling the watchdog timer and the failover monitor

The watchdog timer feature is configured in the kernel by default. Its appearance in the system file follows:

```
# Watchdog Timer
#
# The supported models are:
#
# wdt:  integrated watchdog timer
#
#                                Streams
# Name          Restrictions     Concurrency
# Prefix        Flags            Set
  -------       ------------     -----------
  wdt            o               default
#
```

To enable this feature, ensure that the kernel contains the **wdt()** pseudo device and set up the failover monitor for the software watchdog timer as explained in *Achieving High Availability on AViiON® Systems*.

Setting parameters for use with the watchdog timer

Table 7–5 lists the parameters you can set when using the watchdog timer, and the tool you use to set each parameter: the **dg_sysctl** command, **sysadm**, or SCM.

Table 7–5 Settings for quick recovery from system hangs

Parameter	Value or Path	Where To Set
Autoboot	**auto**	**dg_sysctl** command
Bootpath	**sd(ncsc(),0)root:/dgux -3**	**dg_sysctl** command
Autodump	**auto**	**dg_sysctl** command
Dump device	**ldm_dump(sd(ncsc(),1),sys_dump)**	**dg_sysctl** command
Dump level	**kernel**	**dg_sysctl** command
Power off state	**N/A**	**dg_sysctl** command
Auto–reboot/Boot on Error	**enabled**	SCM
Default boot path	**sd(ncsc(),0)root:/dgux -3**	SCM
Watch dog timer (wtd) configured	`System -> Kernel -> Build`	**sysadm**
Failover monitor set up	`Availability -> Disk Failover -> Alternate Paths -> Add`	**sysadm**

All values set through the **dg_sysctl** command and the SCM must be consistent. For example, if you specified an automatic reboot through the **dg_sysctl** command, you must also choose an auto reboot through the SCM.

Recovering from a system failure

There are three kinds of system failures: power failures, hangs, and panics. The following sections describe how to recover from these failures.

Power failure

Following a power failure, as soon as power returns to the system, it will reboot without operator intervention. You should check to see if the power failure damaged file systems as described in "Restoring file systems after a system failure." If your system has the Uninterruptible Power Supply (UPS) subsystem, see "Avoiding power failure interruptions with the UPS subsystem."

Hang

A hang occurs when an undetected condition causes system activity to halt, effectively freezing every process on the system. You regain control of the system by entering the hot-key sequence at the system console. To type the hot-key sequence hold down the Ctrl key while typing the following series of six bracket characters:

```
] [ ] [ ] [
```

Another way to express the same series is like this:

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

This sequence induces a panic condition. See "Responding to a panic" in the next section to recover from a panic.

If the hot-key sequence fails to induce a panic, use the hardware reset switch to interrupt the hang. Resetting the hardware restores control to the SCM, where you can reboot the system or, if you want the Data General Customer Support Center to investigate the cause of the problem, proceed to make a tape containing information required for diagnosis. See "Completing the diagnostic tape."

If the reset switch fails to restore control to the SCM, turn off power to the computer and power the system back up. Then you should check to see if the hang damaged any file systems as described in "Restoring file systems after a system failure."

Panic

A panic is a condition detected by the kernel that indicates a fatal malfunction or internal software inconsistency. Upon detecting a panic condition, the kernel halts all activity on the system and displays a message like the following at the system console:

```
DG/UX System Panic.  Panic code  57000072
```

If you wish to know the nature of the panic after you have restored the system, see the files in **/usr/release** that list panic codes. For example, to find the file containing information on panic code 57000072, you would execute the following commands:

```
# cd /usr/release ↵
# grep 57000072 *panic.codes ↵
```

You then read the indicated file using a command such as **view** or **more**, both of which offer commands for locating the desired text. See the **view**(1) or **more**(1) manual page.

Workstations that do not have error-correcting memory may occasionally experience an Unrecoverable Memory Error panic, code 1000037. This panic does not necessarily indicate that you have a bad memory module; it simply occurs occasionally on systems without error-correcting memory. Rectify the problem by turning the workstation off and waiting at least one minute before powering it back up and rebooting. If the panic occurs often (more than once every few months), you should call your Data General representative and have your system's memory modules tested and replaced if needed.

Responding to a panic

The first thing to do when a panic occurs is to record the panic code number in the system log. Depending on how you have configured your system, the system may have proceeded beyond the panic code report. In any case, you then have several options:

- You may dump system memory to tape or disk so that the Data General Customer Support Center can help you diagnose the problem.
- You may halt the system, leaving it at the SCM.
- You may reboot the system.
- You may shut off power to the system (can occur without operator intervention on selected computers).

By default, the system responds to a panic by displaying the panic code and then prompting you to take a system memory dump. If you choose to take the system dump, the system issues several prompts for information before starting the dump. After the dump completes, or if you entered **no** to the dump prompt, the system restores control to the SCM. You may then reboot the system. If you wish, you can change any of the default system behavior described here using the **dg_sysctl**(1M) command.

The following sections discuss these options and tell how you can use the **dg_sysctl** command to configure your system to respond to a panic in a variety of ways, including recovering completely without operator intervention. If you do not wish to make a dump tape for diagnosis, see the following section.

Halting the system with no dump

IMPORTANT: We recommend that you take a system dump after every failure resulting from the DG/UX system software and, after copying the system dump and kernel image to tape, submit the tape to Data General for diagnosis. This information is very important to the Customer Support Center in their diagnosis and resolution of your problem.

To halt your system immediately after a panic without taking a system dump, enter **no** at the prompt to take a system dump. If you take the default (**yes**) instead, the system by default takes the system dump and halts. You may then boot with the SCM **BOOT** command.

Using the **dg_sysctl** command's **-r** option, you can set up your system to halt after a panic or system dump without operator intervention. To set up your system to halt after a panic without taking a system dump, issue the following command:

```
# dg_sysctl -d skip -r halt ↵
```

To set up your system to take the system dump without operator intervention and then halt, issue the following command:

```
# dg_sysctl -d auto -f "st(cisc(),5)" -r halt ↵
```

Taking a system dump

To investigate the cause of a panic or hang, the Data General Customer Support Center requires a tape containing at least two files with the following contents:

file 0	The system memory contents, called the *system dump*.
file 1	The kernel executable, typically **/dgux**.

You may also append other files to the tape if you suspect that they may have contributed to the failure. Once you have restored the system, you complete the tape by dumping your kernel executable (and other relevant files, if any) to the tape. The section "Completing the diagnostic tape" tells how to make the tape for diagnostics.

Setting up an automatic system dump

A system dump is either **automatic** or **interactive**. An automatic dump occurs if you have used the **-d** option of **dg_sysctl** to set the the DG/UX automatic dump feature. This feature allows the system to initiate the dump sequence after a panic without operator intervention. The following command line enables the automatic dump feature:

```
# dg_sysctl  -d  auto ↵
```

Setting your system for automatic dump allows you to reduce the recovery time after a panic, but it also implies that you must make sure that the destination dump device is ready at all times to receive the system dump in the event of a panic. A later section describes the dump destination device in more detail.

Setting up an interactive system dump

An interactive dump is your system's default response to a panic. You can set this behavior explicitly by issuing the following command line:

```
# dg_sysctl  -d  ask ↵
```

When the system is configured to perform an interactive dump, the system responds to a panic by displaying the panic code and then the following prompt:

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

If you press Enter, the system prompts for the dump type and the dump destination device, described in the following sections.

Starting a dump from the SCM

On systems where you had to respond to a hang by pressing the hardware reset switch, you can initiate an interactive dump by entering the following commands at the SCM prompt:

```
SCM> reset ↵
SCM> start 1000 ↵
```

The **reset** command completes the system reset; the **start 1000** command starts the dump. The system then begins prompting for the required information.

IMPORTANT: The **start 1000** command does not produce a useful system dump if you have turned off power since the panic occurred or have already produced a system dump with **start 1000** since the panic occurred. In either of these cases, you cannot produce a useful tape for diagnosis, and you may proceed to reboot your system. See "Rebooting after a system failure."

Selecting a dump type and destination device

The dump procedure requires that you decide what type of dump to take and to which device to write the dump.

Selecting a dump type

You may dump either the entire memory of your system or just the kernel memory. A dump of just kernel memory is sufficient to diagnose a hang or panic unless your Data General representative tells you otherwise. The kernel memory dump, which is the default type, is faster and only requires around half the space of a complete dump. To change the default dump type to a complete dump, use the **dg_sysctl** command with the **-l** option:

```
# dg_sysctl  -l  all ↵
```

To restore the dump type to the default, substitute **kernel** for **all** in the command line above.

Selecting a dump destination device

The dump destination is the tape device, virtual disk, or network interface to which you wish to write the dump. By default, the device is the value of the DUMP tunable parameter configured in your kernel. You can override the DUMP parameter setting with the **dg_sysctl** command's **-f** option, for example:

```
# dg_sysctl  -f  "st(ncsc(),5)" ↵
```

You may combine the **–d** and **–f** options to set the automatic dump feature and the dump destination device in a single command line, for example:

```
# dg_sysctl  -d  auto  -f  "st(cisc(),4)" ↵
```

Dumping from an OS client

OS client systems should dump to their network interface, **inen()**. The OS server receives the dump over the network and writes it to a file created for the purpose by the **sysadm** operation Client –> OS –> Add. By default, the Client–> OS–> Add operation creates an empty dump file for the client called **/srv/dump/***client_name*, which it lists in the client's **/etc/bootparams** entry on the server. The operation also exports the dump file for **root** access by the client, adding the appropriate entry to the server's **/etc/exports** file. You should verify that the file system containing the dump file on the OS server has sufficient free space. Use the **df**(1M) command to display the amount of free space in a file system. For information on OS client setup, see Chapter 18.

Dumping to a virtual disk

A system with a local disk can dump to a local virtual disk instead of a tape. The advantage of dumping to disk is that it is faster than dumping to tape, resulting in decreased down-time. The disadvantage is that you must reserve for the purpose a virtual disk large enough to contain the system dump image. The dump image is equal to the size of the computer's physical memory plus 5 percent (if a complete dump), or around half that size (if a kernel dump).

IMPORTANT: You can dump to a virtual disk only if it resides entirely on a local SCSI disk. You cannot dump to a virtual disk that comprises multiple partitions spanning multiple physical disks or to any virtual disk residing on an SMD or ESDI physical disk.

Create the virtual disk with the **sysadm** operation Device–> Disk–> Virtual–> Create. It is a good idea to give the virtual disk an appropriate name such as **sys_dump**. The dump process will write over any data, such as a file system, that resides on the virtual disk at the time of the dump. Therefore, you should not create a file system on the disk and attempt to use it for any purpose other than to contain the dump. By default, the system displays a prompt at the system console before writing to the dump virtual disk, allowing you to specify a different virtual disk or a tape drive if preferred. When the system panic procedure prompts you for the dump

destination device, specify the physical and virtual disks using the following syntax:

vdm_dump(*physical_disk_name*,***virtual***_*disk_name*)

For example, use the **dg_sysctl** command and the **-f** option to set the dump destination device to virtual disk **sys_dump** on local physical disk **sd(cisc(),1)**:

```
# dg_sysctl -f "vdm_dump(sd(cisc(),1),sys_dump)"
```

Rebooting after a system failure

AV8500 and AV9500 systems have a "re-powerup on panic" feature, which determines whether or not the system will be booted automatically following a panic. Enabled by default, this feature overrides the **dg_sysctl** and SCM autoboot settings for the action to be taken after a system failure.

For all other systems, following a system failure, the processor is halted and control is restored to the SCM by default. From the SCM you may then reboot the system using the SCM **BOOT** command.

To minimize your system's downtime, you may set up your system to reboot without operator intervention. Set the automatic boot feature using two operations: the **dg_sysctl** command **-r** option and the SCM's "Auto-reboot/Boot on error" flag, as follows:

```
# dg_sysctl -r auto

SCM> f

From the main menu, select "Change default boot paths" and
set the menu item "Auto-reboot/Boot on error" to
"Enabled."
```

These settings configure your system to reboot whether or not a system dump is taken. By default, the automatic boot feature reboots the system using the most recently used **BOOT** command line, the one last used before the panic occurred. For example, if you last booted **sd(cisc(),0)root:/dgux.test**, the system will reboot using this boot path. To override this default behavior, use both the **dg_sysctl** command with the **-b** option and the SCM "Change default boot paths" menu to specify a different boot path. For example, both the following **dg_sysctl** command line and a matching boot path selected through the SCM "Change default boot paths" option, sets up the system to take a system dump and then boot from **sd(cisc(),2)root:/dgux**:

```
# dg_sysctl -d auto -f "st(cisc(),5)" -r auto -b
"sd(cisc(),2)root:/dgux"
```

You can set up your system to send you mail every time it reboots. This capability is particularly helpful because it reports reboots that occurred in your absence. To enable this feature, edit the **/etc/dgux.params** file. Set the **reboot_notify_START** parameter to true, and set the **reboot_notify_ARG** parameter to one or more local mail addresses. A notification message is sent to each user specified by the **reboot_notify_ARG** parameter each time the system boots.

Completing a diagnostic tape for STR submission

Earlier in the recovery process, you may have decided to take a system dump so that you can make a tape for diagnosis by the Data General Customer Support Center. You submit this tape to Data General along with a Software Trouble Report (STR) explaining the problem. For how to prepare and submit an STR, see the on-line DG/UX release notice in **/usr/release**.

This section tells how to finish preparing the diagnostic tape.

The diagnostic tape needs to be in the following format:

file 0	The system memory contents, called the *system dump*
file 1	The kernel executable, typically **/dgux**

The tape may also include other files if you suspect that they may have contributed to the failure.

IMPORTANT: The diagnostic tape *must* include both the system dump and the kernel. The tape is useless without both of these images.

At this point in the recovery process, you need to copy the system dump to tape as file 0 from one of the following three places.

Dumping directly onto tape

If you wrote the system dump to tape after the failure, it is already on the tape as file 0.

Transferring the dump to tape from a virtual disk

If you wrote the system dump to a virtual disk, transfer the system dump to a tape using the **lsd**(1M) command. For example, if your

virtual disk is named **sys_dump**, use the following command line to dump it to the tape at **/dev/rmt/0**:

```
# lsd -t /dev/rdsk/sys_dump /dev/rmt/0n ↵
```

The command line above does not rewind the tape after writing to it.

Transferring the dump to tape from a file on the OS server

IMPORTANT: This procedure applies to OS clients only.

By default, the Client–> OS–> Add operation creates **/srv/dump/***client_name* as the destination for a system dump by client *client_name*. If you specified a different dump destination when adding the OS client, look there instead. An OS client dumps to the file named in its **/etc/bootparams** entry on the OS server. To transfer the dump file to a tape, use the **cpio**(1) command with the **–oBcv** options. For example, to write a system dump file from OS client **ralph** to the tape at **/dev/rmt/0**, execute the following commands:

```
# cd /srv/dump ↵
# echo ralph | cpio -oBcv > /dev/rmt/0 ↵
```

You should dump the file from the directory where it is located using a relative path name as shown above. Do not dump the file by passing an absolute pathname to **cpio**.

Dumping the kernel executable

With the system dump written to tape, you are ready to dump the kernel executable. Be careful not to overwrite the system dump when you dump the kernel; if you overwrite the system dump, the Data General Customer Support Center cannot diagnose your problem. To make sure the tape is positioned at the end of the system dump (file 0), use the **mt**(1) command to position the tape. For example, to position the tape in tape drive **/dev/rmt/0** at the end of file 0, issue the following commands:

```
# mt -f /dev/rmt/0 rewind ↵
# mt -f /dev/rmt/0n fsf 1 ↵
```

If there is not room on the tape for the kernel executable, you may write it to a second tape instead.

By default, the kernel executable is **/dgux**. If the failure occurred with a different kernel, however, you should dump it instead of **/dgux**. If dumping an OS client's kernel from the OS server, be careful to dump the correct file. The file that appears as **/dgux** in an OS client's file system appears on the OS server as **/srv/release/***release_name*/**root/***client_name*/**dgux**. For example, if you want to dump the kernel that OS client **ralph**, who uses the primary release, refers to as **/dgux**, you should dump **/srv/release/PRIMARY/root/ralph/dgux**.

To dump the kernel, use the **cpio**(1) command with the **–oBcv** options. For example, the following command line dumps **/dgux** to the tape at **/dev/rmt/0**, rewinding the tape when done:

```
# cd / ↵
# echo dgux | cpio -oBcv > /dev/rmt/0 ↵
```

You should dump the file from the directory where it is located using a relative pathname as shown above. Do not dump the file by passing an absolute pathname to **cpio**.

Dumping other files to tape

If you suspect that other programs or files may have contributed to your system's failure, you may include them on the tape as well. You may dump these files as tape files 2, 3, and so on. As when dumping the kernel executable, dump them using **cpio –oBcv** using relative path names.

Labeling the tape

When you have finished making the tape, label it specifying the name of your company, the cause of the failure, the date, and the contents of the tape. Your label might look like this:

```
BLUE DAEMON SYSTEMS, INC., Durham, NC
Panic code:  3400002
Date:  April 6, 1993
File 0:  system memory dump
File 1:  kernel executable
File 2:  miscellaneous files
Density:  QIC-525 tape at high density
cpio format:  cpio -oBcv
```

Restoring file systems after a system failure

A failure on a DG/UX system may not damage files on your system. Damage does occasionally, occur, however, resulting in file system metadata becoming inconsistent or data being lost. The first time you boot your system after a failure, the DG/UX system performs several operations intended to seek out and, where possible, repair damage to files and file systems.

By default, the system invokes **fsck** to check the / file system upon rebooting, both after a system failure and during normal operation. If you do not want this initial check to occur, or if you want to change the kind of check that **fsck** performs, change the RUNFSCK and FSCKFLAGS tunable parameters in the system file you use to build your kernel. For explanations of these parameters, see the **/usr/etc/master.d/dgux** file.

If the system failure damaged files necessary for bringing up the system, **fsck** may fail. If this happens, see "Repairing damaged DG/UX system files."

Once the system has booted successfully, it will proceed to check local file systems according to their **fsck** pass numbers, which you may review with the **sysadm** operation `File System-> Local Filesys-> List`. You can speed up this process by mounting your file systems for fast recovery. For information on fast recovery file systems and **fsck**, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

When file system checking is complete, you should check **/etc/log/fsck.log** and all local file systems' **lost+found** directories to see if you need to restore any lost or damaged files from backup. To restore files from backup, see Chapter 8.

The **fsck** utility has no way of verifying the contents of files on your system. If you use a database product for example or some other software that can check the files it uses, you may want to invoke them for this purpose. The **fsck** utility can verify only the structure of the DG/UX file system.

Repairing damaged DG/UX system files

This section describes how to recover after **fsck**(1M) fails to repair the / or **/usr** file system at boot. As a result, the system will not boot. One common error that makes it impossible to boot a system is when the **/etc/inittab** file is damaged. When this happens, you frequently see a message like this when **init** starts running:

```
... SAD autopush configuration failed ...
```

If a system or disk failure damages **inittab** or other DG/UX system files (those in the **/usr** or / file systems), you need to repair the file systems and restore the damaged files from backups. If you cannot repair the file system, you need to reload the system software from either the release media or a bootable copy of the file systems made with the **systemtape** utility.

If you are running DG/UX 5.4R3.00 or later, you can use the CD–ROM release disk to repair your system. The CD–ROM can be booted into repair mode, under which a complete working environment is available from file systems on the CD. To boot the CD into repair mode, specify the **-R** option flag in the SCM boot command line:

```
SCM> b sd(cisc(),3) -R
```

After you have the repair mode environment running, repair the damaged / and **/usr** file systems' virtual disks with the **fsck** command and then mount them on different mount point directories. You can then copy files from the CD or from system backups to repair any damage.

If you do not have the CD-ROM release, you can use stand–alone **sysadm** to repair the file systems. Boot stand-alone **sysadm**, either from **/usr** or the release tape.

If you want to boot from **/usr**, use a command line like the following, where you specify the physical disk on which the **/usr** file system resides:

```
SCM> b sd(cisc(),0)usr:/stand/sysadm ↵
```

An attempt to boot stand-alone **sysadm** will fail if the **/usr** file system is corrupt or if the **/usr** file system is built on a virtual disk consisting of multiple partitions spanning multiple physical disks. If stand-alone **sysadm** fails or has been deleted from **/usr**, then boot the release tape for the revision your system is running using a command line like the following:

```
SCM> b st(cisc(),4) ↵
```

If stand-alone **sysadm** boots successfully, go to the File System Management Menu and use the Check a File System operation to check the / and **/usr** file systems. When prompted for **fsck** flags, specify **-xlp**.

If you still cannot boot the system, use the Check a File System operation again, but specify the **-y** option for **fsck**. The **-y** option

repairs all non-fatal flaws in the file system, even if the repair results in lost files or data.

If **fsck -y** fails, use the stand–alone **sysadm** installation procedure to get the damaged / and **/usr** files systems mounted. Execute Install Software–> Prepare Virtual disks. Answer the system queries as follows:

```
2.  Prepare required virtual disks

Run this step now?  [yes] ↵
Required File System Mount Points:

     File System    Virtual  Current   Action     Blocks  Physical
     Mount Point    Disk     Blocks    Required   To Add  Disk
     ------------  --------  -------   --------   ------   --------
     -none-         swap      50000      None        -    sd(insc(0),0,0)
     /              root      40000      None        -    sd(insc(0),0,0)
     /usr           usr       240000     None        -    sd(insc(0),0,0)

Modify this information? [no] ↵
```

At this point, escape to the shell by entering **!** at the **sysadm** prompt. The damaged / file system is mounted as **/mnt/root**. the damaged **/usr** file system is mounted as **/mnt/usr**. Set your path as follows to gain access to the normal commands:

```
# PATH=/mnt/root/sbin:/mnt/usr/bin:/mnt/root:sbin↵
# export PATH ↵
```

Note that some commands may not work if they use shared libraries, since the real **/usr** file system that contains them is mounted in a different place than normal. You can now change directory to **/mnt/root** and **/mnt/usr** to examine and repair the damaged file systems. If you are repairing a damaged **inittab** file, there may be an undamaged backup of it in **/etc/inittab.backup**. The original prototype file is in **/etc/inittab.proto**.

When you finish fixing the file systems, exit the shell by entering **exit**. This puts you back in **sysadm**. Exit **sysadm** by entering **q**. That shuts the system down. You can now reboot your repaired / and **/usr** to bring the system back up.

If **fsck -y** succeeds, try to boot the system. If the boot fails, you must boot the release tape and reload the DG/UX system files by going to the Install System Software Menu and performing the Load software operation. The Load software operation will load system files as necessary. When it has finished, you may replace any files you wish by restoring them from backup.

If all of these options fail, you must re-install the DG/UX system completely. Do this by going to the Virtual Disk Management Menu and using the Delete a Virtual Disk operation to delete the virtual disks containing the / and **/usr** file systems (**root** and **usr** virtual disks). Then go to the Install System Software Menu and select option 7, All steps. You will also need to re-install any packages that had been installed before the failure.

As an alternative to booting from the release tape, you can use a bootable copy of your / and **/usr** file systems made with the **systemtape** utility. In addition to normally scheduled backups, you can use **systemtape** to create a bootable tape of these file systems at regular intervals. You can then use that tape if you need to restore your system. See the **systemtape**(1M) manual page for more information.

Avoiding power failure interruptions with the UPS subsystem

IMPORTANT: This section applies only if you have installed the Uninterruptible Power Supply (UPS) subsystem on your computer.

The UPS subsystem monitors the power supply to which your system is connected. If the power supply fails, the UPS subsystem provides a limited additional power supply. Depending on how you configure the UPS daemon running on the host, it may perform shutdown and/or reboot of the host depending on the state of the power supply and the UPS backup battery.

Once set up, the UPS daemon starts at boot and polls the backup battery unit at regular intervals (30 seconds by default). The daemon polls for two pieces of information: the status of the line power supply, and the status of the backup battery. This is how the system functions under normal circumstances.

When line power fails, the UPS backup battery supplies power to the host. The next time the UPS daemon polls the backup battery unit, it detects that line power has failed and does the following on the host:

1. The UPS daemon logs a system error using the **syslog** error logging facility.

   By default, **syslog** responds to the UPS message by alerting all users of the power failure. You can change this behavior by editing **/etc/syslog.conf** and **/usr/sbin/ups.action**. See **syslog.conf**(5) for more information.

   The UPS daemon invokes **/usr/sbin/ups.action** under either of these conditions:

- Line power has terminated.
- Line power returns before the battery dies.
- Battery power is low.
- Line power returns when the system is on battery power.

This script uses the **wall**(1M) command to send commands to all users. If you do not want to notify all users, you may edit the script to redirect messages to the system console or elsewhere.

2. The UPS daemon begins counting down a time-out sequence before shutting down the system.

   You may define the length of the time-out sequence using a **sysadm** operation to be discussed later.

3. If line power returns before the UPS daemon begins the shutdown, the daemon aborts the time-out and the backup battery unit restores the normal line power supply to the computer. If line power does not return before the time-out ends, however, the UPS daemon shuts the system down to single-user mode.

   The system remains in single-user mode either until line power returns or until the battery fails and the system powers down. If line power returns before the battery fails, the UPS daemon takes the system back up to the default run level. If line power returns after the battery has failed, the system reboots.

   The operations for managing the UPS subsystem are in the **sysadm** menu `Device-> UPS` and `Availability-> UPS`. These operations call the **admups**(1M) command to perform the requested operation. The **admups** command may also offer additional functions.

Setting up the UPS subsystem

To use the UPS subsystem, you need to dedicate a terminal line for connection to the UPS backup battery unit. The UPS daemon running on the host uses the terminal line to communicate with the backup battery unit. Connect the UPS backup battery subsystem to the port using the Data General cable supplied specifically for this purpose.

The serial port must include modem control. The serial port must not have a port service currently assigned to it. It is not sufficient simply to disable the port service on the port; you must delete the port service with the **sysadm** operation `Device-> Port-> Port Service-> Delete`. For information on port services, see Chapters 10 and 11.

IMPORTANT: If you start a port service on the port selected for the UPS system, unpredictable results could occur, possibly requiring sudden shutdown of the system.

To set up the UPS subsystem, connect the UPS power cables according to the UPS hardware documentation. Connect the communication cable to the UPS backup battery unit and the selected serial port on the computer host. It is important that you complete installation of the UPS subsystem before starting the UPS daemon.

IMPORTANT: If you start the UPS daemon before you have installed the UPS backup battery unit, unpredictable results could occur.

Execute the **sysadm** operation `Device-> UP -> Start` or `Availability-> UPS-> Start`. This operation starts the UPS daemon and sets up your system to start the daemon at boot. The operation prompts for the following information:

`Polling Interval`
This parameter determines how often, in seconds, the UPS daemon polls the backup battery unit for the status of line power and backup battery viability. In effect, this value determines the maximum number of seconds that may pass between a line power failure and the beginning of the time-out countdown initiated by the UPS daemon. By default, the UPS daemon polls the backup battery unit every **30** seconds.

`Timeout`
This parameter determines how many seconds the UPS daemon waits before commencing shutdown after detecting that line power has failed. If the parameter is **0**, the UPS daemon waits indefinitely, delaying the shutdown until it detects that the backup battery unit is low on power. The backup battery unit is considered low on power when it has approximately 2 minutes of power left. By default, the parameter is set to **0**.

`Serial Port`
This parameter is the pathname of the serial port that the UPS daemon will use to communicate with the backup battery unit. The port must include modem control. There should be no port service, whether enabled or disabled, associated with this port. See Chapters 10 and 11 for information on port services. Setting this parameter to **none** effectively disables the UPS daemon and makes the port available for use with a terminal, modem, or printer.

Stopping the UPS subsystem

To stop the UPS daemon and change your system setup so that the UPS daemon no longer starts at system boot, select the operation `Device-> UPS-> Stop`.

▶ To restore use of the dedicated serial port as a normal terminal line, follow these steps:

1. Use the **sysadm** operation `Device-> UPS-> Parameters-> Set` to reset the daemon's serial port parameter to another port or to the value **none**. Setting the port to **none** stops the daemon and disables it from starting at boot.
2. Remove the specialized UPS cable connected to the port. Refer to the UPS hardware documentation before disconnecting the UPS backup battery unit.
3. Use the **sysadm** operation `Device-> Port-> Terminal-> Add` to start the login service for the serial port.

Setting and displaying UPS parameters

The **sysadm** menu `Device-> UP -> Parameters` contains the Set operation for setting the values of the UPS parameters. You do not need to stop the UPS daemon to change its operating parameters. Use the List operation to display the values of the parameters. For a discussion of the parameters, see "Setting up the UPS subsystem."

Displaying UPS status

Use the **sysadm** operation `Device-> UPS-> List` to display the status of the UPS subsystem. The display indicates:

- Whether the UPS daemon has detected a line power failure.
- Whether the backup battery unit has reported that it is approaching failure.
- Whether the UPS daemon has initiated the shutdown sequence.

Optionally, you can also use the operation to review the history of UPS events as appearing in the system log maintained by the **syslog** system error logger.

End of Chapter

8 Backing up and restoring files and file systems

The backup utilities provide a means of saving file systems and restoring files or file systems when needed, such as when recovering after a failure. Typically, you perform a backup operation every evening (or during some off hour), copying to tape any files that have changed since the last backup.

There are several types of backups: monthly (full), weekly, and daily. The default backup cycle defines a one-month schedule for performing backups that involves all three types of backup. First you perform a monthly backup, which copies every file on the system onto the backup tape. Then during the next week, you take daily backups at the end of each work day except Friday. A daily backup copies files changed during the previous day. On Friday, you take a weekly backup, which copies all files changed during the previous week. You continue this pattern for four or five weeks, until the beginning of the next month, when you restart the cycle with another monthly backup.

The reason the backup cycle is such a complicated mixture of backup types is because it makes both file backup and file restoration easier. Consider instead a system where you back up every file, whether it has changed or not, every night. The nightly backup operation would take a lot of time and magnetic media. In addition, restoring a single file would require that you search through the previous night's backup of the entire system. Consider also the opposite scenario, where you make one monthly backup and then supplement it with thirty days of daily backups, for example. The daily backup takes less time, but restoring a file can be complicated and time-consuming if you do not remember the exact day when the file was last changed.

In addition to the backup procedures covered in the following sections, you might consider making a bootable tape of your / and **/usr** file systems periodically with the **systemtape** utility. Such a tape can help you recover your system after a failure. See the **systemtape**(1M) manual page for more information.

The Legato NetWorker is also available to manage your backups. See the *Legato NetWorker Administrator's Guide* for more information.

Backing up files

You back up systems using the **sysadm** operation `File System-> Backup-> Create` operation or the commands **dump2** and **cpio**. The **sysadm** operation invokes **admbackup**(1M), which invokes the **dump2**(1M) command.

The following sections mention dump levels and dump cycles, terms referring to the schemes used to coordinate dump tapes and the depth of file system changes dumped. For a complete discussion of dump levels and dump cycles, see "Managing the backup cycle."

Backing up a file system with sysadm

To make a backup of a file system, select the **sysadm** operation `File System-> Backup-> Create`. This operation invokes the **admbackup**(1M) command.

The Create operation checks your current position in the backup cycle to see what kind of backup to make: monthly, weekly, or daily. The operation then scans **/etc/fstab** to locate active file systems whose backup-type field matches the backup type scheduled for today according to the backup cycle. The operation then calls the **dump2**(1M) program for each file system that is scheduled to be backed up.

The Create operation prompts you for the following information.

`File System(s)`
: Select the file systems for backing up. Specify **all** to back up all file systems whose backup type (as it appears in **fstab**) matches the backup type for this backup (as indicated by the current position in the backup cycle).

  The **dump2**(1M) command, which performs the backup, does not traverse mount points when backing up a file system. This means that in the process of backing up one file system, it will not also back up another file system mounted within the first. For example, while backing up **/usr**, **dump2** will not also back up **/usr/opt/X11**. If you want to back up **/usr/opt/X11** in addition to **/usr**, you must specify **/usr/opt/X11** explicitly.

  You can back up only local file systems.

`Device`
: Specify the tape or other device to use for the backup. For example, **/dev/rmt/0**.

`Medium Type`
: Specify the type of medium (such as tape) for the backup

device. The available types correspond to the various kinds of tapes and other media and the data densities they support. To get a listing of the available media, use the **sysadm** operation `File System-> Backup-> Medium-> List`.

The List operation gets its information from the readable file **/etc/dumptab**. The default medium type is a QIC-150 cartridge tape. To change the default medium type, use the **sysadm** operation `File System-> Backup-> Medium-> Default`.

`Pack onto One Tape or Medium`

By default, the system packs a backup tape with as many file systems as possible, to conserve tape. When the tape is full, the system requests that you mount a new tape so it can continue with the backup. As an alternative, this query lets you choose to start each file system backup on a new tape. When the system finishes backing up the file system, it requests that you mount a new tape before starting on the next file system.

`Update Databases`

Select this feature to force the operation to change the backup databases to reflect that this backup occurred. If you select **all** in the `File System(s)` query, the operation updates the backup databases. If you do not select **all** file systems for the backup, the operation leaves the databases unchanged.

`Additional dump2 Options`

Specify any other options for the **dump2** command that the operation uses to perform the backup. You can override the backup level indicated in the backup cycle by specifying a different level as an additional **dump2** option. For example, to specify a full backup of the system, specify the option **-0** (0, zero, represents a full backup). See the **dump2**(1M) manual page for more information.

Backing up files from a virtual disk

The **dump2** program copies some or all files on a virtual disk to the backup medium based on the backup level. There are 10 levels: 0 through 9. Execute the **dump2** program by specifying a virtual disk and a backup level as in the following:

```
# /usr/sbin/dump2 -0uf /dev/rmt/0 /dev/dsk/root ↵
```

where:

`0` Specifies backup level **0**.

`u` Updates the **/etc/dumpdates** file.

`f` Specifies the backup device pathname.

The **dump2**(1M) manual page lists all available options.

The backup level number instructs **dump2** to make a copy of each file that has been modified since the most recent backup at any lower backup level number. For example, if the backup level is 3, **dump2** will make a copy of any file that has been modified since the most recent level 0, 1, or 2 backups. Level 0 backs up every file in the file system because there is no lower backup level. A level 0 backup is often called a *full* backup. A monthly backup is typically a full backup.

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 details). If either of these is later than the backup time for the file system at the appropriate backup levels, then the file was modified since the previous lower level backups. The **dump2** command knows when the file system was last backed up at any given level because it keeps this information in the file **/etc/dumpdates**. This file contains lines of the form:

```
/dev/rdsk/root        0 Fri Jan 14 23:58:58 1994
```

In the example above, the most recent level 0 backup for **/dev/rdsk/root**, the root file system, was made at 11:58 p.m. on January 14, 1994. An entry is added to **/etc/dumpdates** only after the backup completes successfully. This prevents it from inserting a date for a backup that later aborts. Also, duplicate entries are deleted. In the example above, any other level 0 entries for **/dev/rdsk/root** would be deleted when adding the new one.

Backing up a directory to tape

When you have a small-scale backup 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 backup a directory named **/sales/smith** and its subdirectories and files, do the following:

1. **Mount a tape and put the drive on-line.**

2. **Go to the directory you wish to backup:**

```
# cd /sales/smith ↵
```

3. **Backup everything in the directory to tape:**

```
# find . -print | cpio -ocvB > /dev/rmt/0n ↵
```

where *n* =

n No rewind

l Low density

u Uncompressed

The contents of **/sales/smith** have been backed up 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.

Backing up individual files to tape

▶ 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 backs up the contents of all three files.

We directed the output of the backup to raw magnetic tape (**rmt**), device 0.

Restoring files

You can restore files using either the **sysadm** operation `File System-> Local Filesys-> Restore` or the **restore**(1M) command. The **sysadm** Restore operation invokes the **admbackup**(1M) command, which invokes the **restore** command; therefore, the two methods are equally reliable.

Restoring files with sysadm

Use the Restore operation to copy files or file systems from a backup tape to disk. This operation is useful for recovering from disk

failures or moving file systems from one disk to another. The Restore operation uses the **restore**(1M) command to retrieve files and file systems from backup tapes created using the **dump2** command.

The following steps outline the simplest, though not necessarily the fastest, method of restoring a file or file system.

1. Before restoring files, make sure the file system where you will restore the files has enough free space to hold them.
2. Retrieve the most recent monthly backup and use it to restore the desired files.
3. Retrieve the weekly backups made since the monthly backup and, loading them in order of earliest first, restore the desired files.
4. Retrieve the daily backups made since the last weekly backup and, loading them in order of earliest first, restore the desired files.

   You can shorten the amount of time needed to restore a file if you know when the file was last modified. For example, if on Thursday you accidentally delete a file that you know you modified the previous Wednesday, you need only load Wednesday evening's daily backup in order to restore the file. If you last modified the file last week some time, you only need to load last Friday's weekly backup to restore the file. When you are restoring a group of files or an entire file system, it is more difficult to determine which backups you need, so you may find it easier just to start from the monthly backup and work your way up from there.

   Once you have loaded the first tape, you can invoke the Restore operation. The operation prompts you for a directory where it should restore the files. This directory must already exist. The operation also lets you choose between full (non-interactive) mode and interactive mode. Full mode restores an entire file system. Interactive mode is best for restoring individual files because it allows you to search through the tape and restore only the files you want. The interactive mode is managed by the **restore** command. Instructions for using the interactive mode of the **restore** command are in "Restoring files interactively."

Restoring file systems with restore(1M)

Restore file systems using the **restore**(1M) command with the **r** option. If a file system (other than / or **/usr**) 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 backup

2. All weekly backups made since the most recent monthly backup
3. All daily backups made since the most recent weekly backup

IMPORTANT: Attempting to restore the **/usr** file system produces an error when the **/usr/sbin/restore** command attempts to overwrite itself while it is being executed. To avoid this problem, copy **/usr/sbin/restore** to **/tmp**, then invoke **/tmp/restore** to restore the **/usr** file system. Alternatively, you may invoke **restore** interactively and select all files in **/usr** except **/usr/sbin/restore** for restoration (see the next section on interactive use). This latter method must be used if you are restoring using **sysadm**.

Consider an example environment where we do our weekly backups on Friday. If the file system is lost on Wednesday of the second week (before the Wednesday backup), we need the following tapes:

Monthly
Weekly (1)
Weekly (2)
Monday (2)
Tuesday (2)

▶ To restore the file system:

1. **Unmount the file system:**

```
# /etc/umount /dev/dsk/foo ↵
```

2. **Remake the file system (be aware that this command destroys all data on the virtual disk):**

```
# /usr/sbin/mkfs /dev/dsk/foo ↵
```

3. **Check the file system:**

```
# /etc/fsck /dev/dsk/foo ↵
```

4. **Mount and restore the monthly tape set:**

```
# /etc/mount /dev/dsk/foo /mount_name ↵
# cd /mount_name ↵
# /usr/sbin/restore rf /dev/rmt/0 ↵
```

5. **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.

Restoring individual files interactively

You use **restore** in interactive mode either by selecting the Interactive Mode option in the **sysadm** Restore operation or by invoking the **restore** command with the **i** (interactive) option. In interactive mode, **restore** issues a prompt and waits for you to enter commands. You can use the following commands:

`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.

▶ With the backup tape of the **/sales/accounts** file system mounted, we follow these steps to restore the file **/sales/accounts/smith/redeye**:

1. **Change to the directory where redeye exists:**

```
restore> cd smith ↵
```

2. **Verify that redeye exists:**

```
restore> ls redeye ↵
redeye
```

3. **Add redeye to the list of files to be extracted:**

```
restore> add redeye ↵
```

4. **List redeye again to verify that it is marked for extraction:**

```
restore> ls redeye ↵
*redeye
```

5. **Perform the extraction:**

```
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 '.'? [y/n] n ↵
```

IMPORTANT: Whenever **restore** asks `Specify next volume #`, always answer "1." This is the correct response, no matter which tape your file is on.

6. **Exit restore:**

```
restore> quit ↵
```

By default, **restore** writes the file to **/tmp** so that you can inspect the file before installing it in its original directory.

After extracting tapes, the operation prompts you to mount the next tape, if desired. You may continue to mount backup tapes and restore files in this manner until you have restored all the files you need.

Managing the backup media

The **sysadm** operation `File System-> Local Filesys-> Medium` provides operations for managing the backup medium table, **/etc/dumptab**, which contains information about the storage media supported for making backups. The medium table shipped with the DG/UX system probably contains all the media entries you need.

An entry in the medium table includes fields for medium name, the block size used for data transfer, the capacity of each medium (such as a tape cartridge), and a description of the medium. The Add and Modify operations of the Medium menu prompt you for values for each of these fields. The Delete operation prompts you only for the medium name. The List operation lists current medium table entries. An example listing follows:

Medium	Block Size	Capacity	Description
default	16	150M	QIC-150 150MB 1/4" Cartridge Tape
pre_5.4	10	150M	Used for restoring pre-5.4 backups
cartridge_150	16	150M	QIC-150 150MB 1/4" Cartridge Tape
cartridge_320	16	320M	QIC-320 320MB 1/4" Cartridge Tape
cartridge_525	16	525M	QIC-525 525MB 1/4" Cartridge Tape
cartridge	16	150M	QIC-150 150MB 1/4" Cartridge Tape
reel_800	16	19M	800 bpi 1/2" Reel-to-Reel Tape
reel_1600	16	38M	1600 bpi 1/2" Reel-to-Reel Tape
reel_6250	16	143M	6250 bpi 1/2" Reel-to-Reel Tape
reel	16	38M	1600 bpi 1/2" Reel-to-Reel Tape
video	16	2200M	8mm 2GB Video Tape
worm_optimem	16	1200M	2458MB Optimem WORM Platter (1 side)
worm	16	1200M	2458MB Optimem WORM Platter (1 side)

The default medium, QIC-150 cartridge, appears as the default medium in the Backup menu's Create operation. You can reset this default to any other entry with the Default operation.

Managing the backup cycle

Use the **sysadm** operation `File System-> Local Filesys-> Cycle` menu to select a backup cycle, set your position within the selected backup cycle, and list the next backup scheduled for the system.

The backup cycle determines the order and types of backups that occur on your system. The three types of backups are:

Monthly (full)
This backup copies every file on the system.

Weekly
This backup copies every file changed since the most recent weekly or monthly backup.

Daily This backup copies every file changed since the most recent daily or weekly backup.

The default backup cycle is intended for systems that have a lot of disk space and where a lot of data changes from day to day. The default backup cycle covers a five-week span starting with a monthly (full) backup and followed by a number of daily and weekly backups. After the monthly backup, each of the five following weeks follows the same pattern: there are four daily backups (Monday through Thursday) and one weekly backup (Friday). At the end of the month, you start the cycle over again.

In addition to the default backup cycle, there are two other backup cycles from which to choose. The medium disk cycle performs a

complete (full) backup every week, with daily backups the other four working days. This backup cycle is intended for systems with a fairly high amount of disk space but where a relatively low portion of data changes frequently.

The third backup cycle is the small disk cycle. This backup cycle involves a full backup every day. This cycle is good for systems whose file systems can all fit on a single tape.

The default backup cycle, the one intended for large systems, follows this scheme of dump levels:

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 backup occurs on the Friday following the Friday when you performed the monthly backup. You could perform these dumps in order by executing the **sysadm** operation `File System-> Backup-> Create` every weekday evening, or you could perform them by invoking **dump2** every weekday evening. For example, if you wanted to back up the root file system according to this cycle, dumping it to tape at **/dev/rmt/0**, you would use the following command lines on successive weekday evenings:

Dump	Command
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**

Continued

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.

The default backup cycle is the file **/etc/sysadm/dumpcycle**. The file looks like this:

```
@[dwm]   0       n       Monthly Set
[d]      6       n       Week 1 - Monday Set
[d]      7       n       Week 1 - Tuesday Set
[d]      8       n       Week 1 - Wednesday Set
[d]      9       n       Week 1 - Thursday Set
[dw]     1       n       Week 1 - Weekly Set
[d]      6       n       Week 2 - Monday Set
[d]      7       n       Week 2 - Tuesday Set
[d]      8       n       Week 2 - Wednesday Set
[d]      9       n       Week 2 - Thursday Set
[dw]     2       n       Week 2 - Weekly Set
[d]      6       n       Week 3 - Monday Set
[d]      7       n       Week 3 - Tuesday Set
[d]      8       n       Week 3 - Wednesday Set
[d]      9       n       Week 3 - Thursday Set
[dw]     3       n       Week 3 - Weekly Set
[d]      6       n       Week 4 - Monday Set
[d]      7       n       Week 4 - Tuesday Set
[d]      8       n       Week 4 - Wednesday Set
[d]      9       n       Week 4 - Thursday Set
[dw]     4       n       Week 4 - Weekly Set
[d]      6       n       Week 5 - Monday Set
[d]      7       n       Week 5 - Tuesday Set
[d]      8       n       Week 5 - Wednesday Set
[d]      9       n       Week 5 - Thursday Set
[dw]     5       n       Week 5 - Weekly Set
```

The columns in the table are:

`Cycle` The first column lists the cycle letters that correspond to those in **/etc/fstab**, which indicate when the file systems will be backed up. For instance, file system **/comm** might be set to `w`, so it is only backed up once per week. The cycle letters are:

`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` The second column shows numbers that are used internally by the **dump2** program. The ones we supply need not be changed for normal system operation.

`Multi` The third column indicates whether multi-dumping shall be in effect for the backup. Multi-dumping means backing up more than one file system per tape. If `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.

`Description`
This column is a comment describing the backup cycle entry. We recommend that you label your tapes so that they match the entries in the backup cycle list. `Monthly` means backup all file systems marked with `d`, `w`, or `m` (daily, weekly, or monthly). `Monday Set` means backup all file systems marked with `d` (daily), and so on with the other weekdays. Friday's backup becomes part of the `Weekly Set` of backup tapes.

The "at" symbol (@) indicates the current position in the backup cycle.

The complete set of backup cycles that provided with the DG/UX system are in the directory **/etc/sysadm/dumpcycles**. This directory also contains descriptions of each backup cycle.

To set a backup cycle for use on your system, select the **sysadm** operation `File System-> Backup-> Cycle-> Select`.

The system keeps track of your current position in the backup cycle. After every backup, the system moves the pointer (indicated by @) ahead to the next backup, advancing line-by-line down the cycle until the end of the month. On the first day of the next month, use the Position operation to reset the pointer to the top of the list to restart the backup cycle.

It is possible for the current pointer position in the list to be wrong if backups are skipped for a day or more. To restore the backup cycle pointer to the correct position, use the Position operation.

To display your current location in the backup cycle, select the List operation.

For a complete description of backup cycle format, see **dumpcycle**(4).

End of Chapter

9 Localizing the DG/UX system

Localizing your DG/UX system means tailoring it for a specific language or locale. This chapter describes how to set the time and date on your DG/UX system and how to localize the system using locale databases and such internationalization features as ISO and 8-bit code sets. The chapter also contains a list of the C library routines provided to support internationalization.

DG/UX system internationalization is based on the UNIX® System V Release 4 Multi-National Language Supplement (MNLS).

Setting the time and date

To set the time and date, use the **sysadm** operation `System-> Date-> Set`. When you set the time, you may specify:

- Day of the month
- Hour
- Minute
- Year
- Time zone

For a time zone, select one of those listed on the screen or in the help message. See the **timezone** manual page for more information on the time zone format.

We recommend that you not set the time back while the system is at run level 1 or higher. Setting the time back while at any level above single-user mode may cause system daemons and X Window System clock clients to behave erratically. Take the system down to single-user mode before setting the time back.

You may set the time forward at any run level without disrupting the function of system daemons.

To display the time and date, use the **sysadm** operation `System-> Date-> Get`. In the shell, you set and display the date and time with the **date** command.

Alternatively, you may prefer to get time from a LAN-based designated clock server, providing the precise time to all hosts on the LAN. Refer to *Managing TCP/IP on the DG/UX™ System* for information to set up NTP (Network Time Protocol) in your network.

Managing native language support

You can customize the system's on-line environment for a particular language, country, or area of a country. Locale databases shipped with the DG/UX system define currency symbols, collating sequences for sorting operations, comma and decimal point usage in numbers, and other symbolic and organizational criteria corresponding to native language usage and style in over 70 geographical regions.

Certain DG/UX commands, including **cp**, **find**, **ln**, **mv**, **rm**, and **tar**, accept native language responses to yes and no questions. For example, if you set the **LANG** variable to a French language locale, these commands accept **oui** and **o** in addition to **yes** and **y**.

The native language support facility also provides support for applications that use the X/Open and USL-style message facility message catalogs. A message catalog is a list of strings in a given language intended for use in a specific program or application. Some applications ship with multiple message catalogs, each one supporting the application in a different language.

Locale environment variables

A locale contains localization data for a particular language, territory, and code set. Each locale has categories that contain information for a specific area of localization. These categories are described in Table 9–1, listed by the name of the environment variable used to control the category.

Table 9–1 Locale environment variables

Variable	Description
`LC_COLLATE`	Collating sequence. Data defining collating sequences to be used for a specific single-byte code set.
`LC_CTYPE`	Character classification and conversion. Data defining character classes for the code set used by the locale.
`LC_MONETARY`	Currency representation. Data defining currency format based on locale conventions.
`LC_NUMERIC`	Numeric representation. Data defining numeric format based on locale conventions.
`LC_TIME`	Date and time format. Data defining date and time display format and language.
`LC_MESSAGES`	USL-style message catalogs. Determines the native-language used to display system error messages stored in USL-style message catalogs. Not recommended for applications that are intended to be portable.

Variable	Description
LANG	Default. LANG is used as the default locale if the corresponding environment variable for a particular category is unset. Also, used to locate X/Open-style message catalogs and is the portable method for locating USL-style message catalogs. The system-wide default value for LANG can be changed with the **sysadm**(1M) command.
NLSPATH	X/Open-style message catalogs. Defines location of native-language system error message catalogs in X/Open format. Uses the value of LANG to determine which locale catalogs are used. The system-wide default value for NLSPATH can be changed with the **sysadm** command.

The language your system uses to display system messages is set using environment variables based on two types of message catalogs, X/Open and USL-style. These catalogs contain system and application messages separate from executable programs. The values of the environmental variables **NLSPATH** and **LANG** are used to locate X/Open message catalogs. The value of **LANG** is used to locate USL message catalogs, although you can override the value of **LANG** by setting **LC_MESSAGES**.

IMPORTANT: Use of **LC_MESSAGES** is not recommended in applications that are intended to be portable. For further details about X/Open and USL message facilities, refer to *Porting and Developing Applications on the DG/UX™ System*.

For other information about each environment variable, refer to the **environ** manual page.

Locales supported on DG/UX

The DG/UX system supports over 70 different locales. A locale in this context is the combination of native language and cultural characteristics commonly understood by people in a particular country or geographical region.

To set up your system to use the locale characteristics defined for one particular locale, you can set the **LANG** variable to a particular locale name. This is convenient when one locale can meet most of your data requirements. After setting **LANG** to a locale name, you can override this setting on an individual category basis by setting the environment variable corresponding to each category to a different locale name.

Locale names have the form:

```
language[_territory[.codeset]]
```

Some locale names are **fr** (French), **fr_CA** (Canadian French), and **fr_CA.850** (Canadian French, using the IBM 850 code set). ISO 8859–1 is the default code set if you omit *.codeset* in the locale name.

The locale names and related information for locales defined on the DG/UX system appear in Table 9–2. The table is organized alphabetically by language except for the first item, the **C** locale definition that you must not delete.

These all appear as subdirectories of **/usr/lib/locale**. You can delete any that you do not need except the **C** locale definitions. The contents of the locale definitions are described in the **setlocale**, **colltbl**, **chrtbl**, **montbl**, and **mkmsgs** manual pages.

Table 9–2 Predefined DG/UX locales

Locale name	Language	Country	Code set	Equivalent name
C	English	United States	ASCII	C–locale
sk	Czech	Czechoslovakia	ISO 8859–2	
da	Danish	Denmark	ISO 8859–1	da_DK
da_DK.850	Danish	Denmark	PC 850	
da_DK.865	Danish	Denmark	PC 865	
nl	Dutch	Netherlands	ISO 8859–1	nl_NL
nl_NL.437	Dutch	Netherlands	PC 437	
nl_NL.850	Dutch	Netherlands	PC 850	
nl_BE	Dutch	Belgium	ISO 8859–1	
nl_BE.437	Dutch	Belgium	PC 437	
nl_BE.850	Dutch	Belgium	PC 850	
en	English	United Kingdom	ISO 8859–1	en_GB
en_GB.437	English	United Kingdom	PC 437	
en_GB.646	English	United Kingdom	ISO 646	
en_GB.850	English	United Kingdom	PC 850	
en_AU	English	Australia	ISO 8859–1	
en_AU.437	English	Australia	PC 437	
en_AU.646	English	Australia	ISO 646	
en_AU.850	English	Australia	PC 850	
en_CA	English	Canada	ISO 8859–1	
en_CA.437	English	Canada	PC 437	
en_CA.646	English	Canada	ISO 646	
en_CA.850	English	Canada	PC 850	
en_US	English	United States	ISO 8859–1	

Continued

Table 9–2 Predefined DG/UX locales

Locale name	Language	Country	Code set	Equivalent name
en_US.437	English	United States	PC 437	
en_US.646	English	United States	ISO 646	
en_US.850	English	United States	PC 850	
fi	Finnish	Finland	ISO 8859–1	fi_FI
fi_FI.437	Finnish	Finland	PC 437	
fi_FI.850	Finnish	Finland	PC 850	
fr	French	France	ISO 8859–1	fr_FR
fr_FR.437	French	France	PC 437	
fr_FR.850	French	France	PC 850	
fr_BE	French	Belgium	ISO 8859–1	
fr_BE.437	French	Belgium	PC 437	
fr_BE.850	French	Belgium	PC 850	
fr_CA	French	Canada	ISO 8859–1	
fr_CA.850	French	Canada	PC 850	
fr_CA.863	French	Canada	PC 863	
fr_CH	French	Switzerland	ISO 8859–1	
fr_CH.437	French	Switzerland	PC 437	
fr_CH.850	French	Switzerland	PC 850	
de	German	Germany	ISO 8859–1	de_DE
de_DE.437	German	Germany	PC 437	
de_DE.850	German	Germany	PC 850	
de_AT	German	Austria	ISO 8859–1	
de_AT.437	German	Austria	PC 437	
de_AT.850	German	Austria	PC 850	
de_CH	German	Switzerland	ISO 8859–1	
de_CH.437	German	Switzerland	PC 437	
de_CH.850	German	Switzerland	PC 850	
el	Greek	Greece	ISO 8859–7	el_GR
is	Icelandic	Iceland	ISO 8859–1	is_IS
is_IS.850	Icelandic	Iceland	PC 850	
it	Italian	Italy	ISO 8859–1	it_IT
it_IT.437	Italian	Italy	PC 437	
it_IT.850	Italian	Italy	PC 850	
it_CH	Italian	Switzerland	ISO 8859–1	
it_CH.437	Italian	Switzerland	PC 437	

Continued

Table 9–2 Predefined DG/UX locales

Locale name	Language	Country	Code set	Equivalent name
it_CH.850	Italian	Switzerland	PC 850	
no	Norwegian	Norway	ISO 8859–1	no_NO
no_NO.850	Norwegian	Norway	PC 850	
no_NO.865	Norwegian	Norway	PC 865	
pl	Polish	Poland	ISO 8859–2	pl_PL
pt	Portuguese	Portugal	ISO 8859–1	pt_PT
pt_PT.850	Portuguese	Portugal	PC 850	
pt_PT.860	Portuguese	Portugal	PC 860	
ru	Russian	Russia	ISO 8859–5	ru_SU
sh	Serbo Croatian	Yugoslavia	ISO 8859–2	sh_YU
es	Spanish	Spain	ISO 8859–1	es_ES
es_ES.437	Spanish	Spain	PC 437	
es_ES.850	Spanish	Spain	PC 850	
sv	Swedish	Sweden	ISO 8859–1	sv_SE
sv_SE.437	Swedish	Sweden	PC 437	
sv_SE.850	Swedish	Sweden	PC 850	
tr	Turkish	Turkey	ISO 8859–3	

Code sets supported on DG/UX

A code set is a collection of characters (visual glyphs) and a way of representing the characters with numeric values; U.S. ASCII (ISO 646) is a well known code set. A code set used by an application program when processing data is called an internal code set. A code set used by a terminal or printer is an external code set.

The DG/UX system supports the following internal code sets:

ISO 8859–1 (Latin 1)
ISO 8859–2 (Latin 2)
ISO 8859–3 (Latin 3)
ISO 8859–4 (Latin 4)
ISO 8859–5 (Latin/Cyrillic)
ISO 8859–7 (Latin/Greek)
PC 437 (United States)
PC 850 (Multilingual)
PC 860 (Portugal)
PC 863 (Canada-French)
PC 865 (Norway)

External code sets supported by the DG/UX system include the internal code sets and the following:

ISO 646 and ISO 646 national variants for the following:
- Denmark
- France
- Great Britain
- Germany
- Italy
- Norway
- Portugal
- Spain
- Sweden
- Yugoslavia

DEC Multilingual
Roman 8
EBCDIC
Data General International (DGI)

The default code set on DG/UX system consoles and X Window System windows is ISO 8859-1, the Western European code set.

Data General terminals and printers support either ISO 8859-1 (or a very close approximation of it) or ASCII. The ISO 8859-1 code set is a superset of ASCII. The locales that specify other code sets are useful only with I/O devices (terminals and printers) that use those code sets.

The DG/UX system provides the most complete support for Latin 1. Character terminals, keyboards, printers, and workstations that support Latin 1 can be used with the DG/UX system. With the exception of U.S. ASCII, each of these code sets requires 8 bits per character. DG/UX commands and libraries properly handle 8-bit characters.

The DG/UX system provides limited support for the DGI and DEC Multilingual code sets. DG/UX utilities transform keyboard input from these code sets to Latin 1. On output to a CRT, DG/UX utilities transform Latin 1 into either DGI or DEC Multilingual. For example, to use a DGI terminal on the DG/UX system, run the script shown in the **codeset_conv** man page.

Setting and viewing native language support variables

The **sysadm** `System-> Language` menu provides operations for setting and listing the native language support parameters for your system. The native language variable, **LANG**, tells the system which locale database to use for determining functionality that varies from region to region throughout the world.

The **NLSPATH** environment variable tells the system what directories to search for message catalogs. By default, the

NLSPATH variable includes directories based on the value of the **LANG** variable. If you install software that loads message catalogs into other directories, you need to add these directories to **NLSPATH**.

Use the Set operation to set your system's global **LANG** and **NLSPATH** variables. Use the Get operation to display the variables' current values.

The following command sets all the locale categories to use data from the locale **fr**, with the exception of **LC_MONETARY**, which uses data from the locale **fr_CH**:

```
LANG=fr; LC_MONETARY=fr_CH; export LANG LC_MONETARY
```

Converting a file from one code set to another

To convert a file from one code set to another, use the **iconv** program. For example, to convert an ISO 8859–1 file named **latin1** to an ASCII file named **ascii** using the best fit available for each character, enter this command:

```
iconv -f 88591 -t ASCII -m b latin1 > ascii
```

For more information about **iconv**, see the **iconv** man page.

Printing a file containing non-ASCII characters

The DG/UX system supports the printing of non-ASCII characters. If you have a line printer that handles the characters you are printing, you can simply use the **lp** command. If you are using a PostScript® printer, you must use the **–S** option on **lp**.

For example, to print a Latin 1 file named **europe1** on a printer whose queue is named **pslaser**, use the following command:

```
# lp -d pslaser -S iso-88591 europe1
```

For more information, see the **lp**(1) and **postprint**(1) man pages.

Displaying non-ASCII characters on a workstation or terminal

To display Latin 1 characters on an AViiON workstation, simply run **mterm** or **xterm** to create a window. To display characters from the Data General International character set, use **mterm –d216** or **mterm –d410**.

To display Latin 1 characters on a terminal, you need a terminal that supports Latin 1. Data General terminal models d217, d413, and d463, or later models in those series support Latin 1 when the terminal is in ISO mode. For more information, see the manual for your terminal.

Entering non-ASCII characters on an ASCII keyboard

The DG/UX system lets you enter non-ASCII characters from an ASCII keyboard. A STREAMS-based **tty** driver and a line discipline module (**att_kbd**) handles single-byte and multi-byte character I/O.

In an **mterm** window, you can enter Latin 1 characters from an ASCII keyboard using the built-in compose keys provided with **mterm**. To view the compose-key sequences, position your mouse cursor on View in the border of the **mterm** window and click the left mouse button. To enter a non-ASCII character, press the Pause key and then type the two-key compose sequence.

At a terminal or in a window, you can enable compose keys by running the following script, in which the ` character is an accent grave.

```
stty_settings=`stty -g`
strchg -p
strchg -p
strchg -h att_kbd
strchg -h ldterm
strchg -h ttcompat
stty $stty_settings
# stty -g bug workaround; set stty values as appropriate:
stty line 1 \
intr \^c erase \^\? kill \^u swtch \^g susp \^n dsusp \^n werase \
\^t -brkint -istrip ixany ixoff echoe echoctl echoke iexten tab0
kbdload /usr/lib/kbd/88591.cpz
kbdset -a 88591.cpz
```

After you have run the script, you can enter all Latin 1 characters from an ASCII keyboard. The compose keys also let you enter certain ASCII characters such as tilde (~) that may not be on all keyboards. To use a compose key sequence, type Ctrl-T followed by the two-character sequence listed in the **cpz** man page.

Creating locale databases

The default location of locale data is **/usr/lib/locale**. The default location of X/Open-style message catalogs is **/usr/lib/nls/msg**. The **locale** directory and the **msg** directory contain locale directories for specific locales.

To create a new locale, create a directory in **/usr/lib/locale** and create files in that directory. Table 9–3 lists the programs you can use to create the database files.

Table 9–3 Programs for creating locale databases

Filename	Command	Description of File Information
LC_COLLATE	`colltbl colltbl.src`	Collating sequence
LC_CTYPE	`wchrtbl chrtbl.src`	Character classification and conversion
LC_MESSAGES/*	`mkmsgs` *catalog*	USL-style message catalogs in USL format
	`gencat -a` *catalog*	X/Open-style message catalogs in USL format
LC_MONETARY	`montbl montbl.src`	Currency representation
LC_NUMERIC	`chrtbl chrtbl.src`	Numeric representation
LC_TIME	`cp time.src LC_TIME`	Date and time format
***.cat**	`gencat` *catalog*	X/Open-style message catalogs in X/Open format

LC_MESSAGES contains a file named **Xopen_info** that contains data not available in **LC_COLLATE**, **LC_CTYPE**, **LC_MONETARY**, and **LC_TIME**. The data in **Xopen_info** comprises the following:

- Default time format
- Default date format
- Default format for date and time
- An equivalent for "yes"
- An equivalent for "no"

While the locale databases specify the language to use for system error messages, the DG/UX system contains message catalogs in only one language, English. The DG/UX system provides several USL-format message catalogs in **/usr/lib/locale/C/LC_MESSAGES** and a few X/Open-format message catalogs in **/usr/lib/nls/msg/C**. Each USL-style catalog begins with **dg** or **ux**. Each X/Open-style catalog ends in **.cat**.

The DG/UX system has utilities for creating and accessing other message catalogs (see Table 9–3), but it does not contain the catalogs themselves. Currently, catalogs of messages in French, German, and Spanish are available in Data General's Western European System Messages Release. For more information about these catalogs, contact your Data General sales representative.

Creating local message catalogs

▶ To create a set of local message catalogs:

1. Copy to a temporary directory all files in **/usr/lib/locale/C/LC_MESSAGES** that begin with **dg** or **ux**.

2. Copy to a temporary directory all files in **/usr/lib/nls/msg/C** that end with **.cat**.
3. Decompile the USL-format catalogs using **mkmsgs**. For example:

```
# mkmsgs -d dgcore > dgcore.txt ↵
# mkmsgs -d uxcore.abi > uxcore.abi.txt ↵
# mkmsgs -d uxlibc > uxlibc.txt ↵
# mkmsgs -d uxlp > uxlp.txt ↵
# mkmsgs -d uxsyserr > uxsyserr.txt ↵
```

4. Decompile the X/Open-format catalogs using **gencat**. For example:

```
# gencat -d catexstr.cat > catexstr.msg ↵
# gencat -d exterr.cat > exterr.msg ↵
```

5. Edit the decompiled source files to change the text from English to the appropriate language. For example:

```
# vi dgcore.txt uxcore.abi.txt uxlibc.txt uxlp.txt
uxsyserr.txt ↵
# vi catexstr.msg exterr.msg ↵
```

   When editing **dgcore.txt**, be sure to avoid deleting blank lines, as this will result in the wrong message being displayed later. Unlike the files whose names begin with **ux**, **dgcore** is an X/Open-style message catalog, even though it is in USL format; i.e., **dgcore** was created by **gencat –a**, not **mkmsgs**. **gencat** cannot decompile files created via **gencat –a**.

6. Recompile the message catalogs. For example:

```
# mkmsgs dgcore.txt dgcore ↵
# mkmsgs uxcore.abi.txt uxcore.abi ↵
# mkmsgs uxlibc.txt uxlibc ↵
# mkmsgs uxsyserr.txt uxsyserr ↵
# gencat exterr.cat exterr.msg ↵
```

7. Install the USL-format catalogs in *locale*/**LC_MESSAGES** and the X/Open-style catalogs in **msg**/*locale*, where *locale* is the name of the locale for which the messages have been translated.

Activating messages for a particular locale

To activate a set of messages for a particular locale, set the environment variable **LANG** or **LC_MESSAGES** to the name of the locale. You can make this the system default by setting it in **/etc/login.csh** and **/etc/profile**. An individual user can make it his

or her default by setting it in **$HOME/.login** or **$HOME/.profile**. A user can also set it for a single shell session.

C library routines that support internationalization

The DG/UX system provides a number of library routines (C functions) to support internationalization. These routines are derived from MNLS. This section lists the man pages that describe these library routines. For information about the routines, see the relevant man page.

addsev
Define additional severities.

curses
CRT screen handling and optimization package.

curs_addwch
Add a **wchar_t** character to a curses window.

curs_addwchstr
Add string of **wchar_t** characters to a curses.

curs_addwstr
Add a string of **wchar_t** characters to a curses window.

curs_getstr
Get character strings from curses terminal keyboard.

curs_getwch
Get (or push back) **wchar_t** characters from curses terminal keyboard.

curs_getwstr
Get **wchar_t** character strings from curses terminal keyboard.

curs_instr
Get a string of characters from a curses window.

curs_inswch
Insert **wchar_t** character before cursor in curses window.

curs_inwch
Get a **wchar_t** character from a curses window.

curs_inwchstr
Get a string of **wchar_t** characters from a curses window.

curs_inwstr
Get a string of **wchar_t** characters from a curses window.

curs_pad
Create and display curses pads.

curs_printw
Print formatted output in curses windows.

curs_scanw
Convert formatted input from a curses window.

getopt
Get option letter from argument vector.

gettxt
Retrieve a text string.

getwc
Get **wchar_t** character from a stream.

getwidth
Get information of supplementary code sets.

getws
Get a **wchar_t** string from a stream.

lfmt
Display error message in standard format and pass to monitor and logger.

mbchar
Multi-byte character conversion.

mbstring
Multi-byte string conversion.

pfmt
Display error message in standard format.

printf
Print formatted output.

putwc
Put **wchar_t** character on a stream.

putws
Put a **wchar_t** string on a stream.

scanf
Convert formatted input.

setcat
Define default catalog.

setlabel
Define the label for **pfmt()** and **lfmt()**.

ungetwc
Push **wchar_t** character back into input stream.

vlfmt
Display error message in standard format and pass to monitor and logger.

vpfmt
Display error message in standard format and pass to monitor and logger.

vprintf
Print formatted output of a variable argument list.

wconv
Translate characters.

wctype
Classify ASCII and supplementary code set.

widec
Multi-byte character I/O routines.

wstring
wchar_t string operations and type transformation.

End of Chapter

10 Terminals, ports, and tty lines

This chapter tells how to manage your system's ports, including terminal lines and the lines you use for UUCP connections. It also explains how to set up terminals to operate in the DG/UX system environment. It assumes the terminal hardware has been installed.

The DG/UX system provides port services through the Service Access Facility (SAF). You can manage port services two ways. One way is by using the operations found in the **sysadm** Device–> Port menu. This chapter covers the operations in the Port menu.

The second way to manage port services is with commands invoked at the shell level. For coverage of this method and the shell commands, see Chapter 11.

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

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

Port tty line numbers

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

`tty`*x*

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

You can add terminals (and printers) to your DG/UX system all at once or one at a time. If you add them all at once, you assign the

same characteristics to all asynchronous ports on your computer. In other words, the RS-232/422 ports on the computer unit, the ports on any Systech asynchronous controllers (VAC/16 controllers), and the ports on the cluster controllers for any Systech asynchronous distributed host adapters (VDA/128 and VDA/255 host adapters) would all have the same characteristics.

If you add terminals one at a time, you may assign different characteristics to different ports.

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

- The tty line number that the DG/UX system assigned to the terminal line controller port where the terminal is connected. This number depends on the terminal controller, number and type of previous terminal controllers, and port number on the controller. If you have ports attached to multiple, different terminal line controllers, you must determine each port's tty line number.

  To learn the tty line numbers, you may need to:

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

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

  c. Find the port where each device is connected

- The type of terminal connected to the port (for example, VT100 terminal or PostScript printer).

  After gathering this information, you can then add the terminals themselves using **sysadm**.

Getting information about controllers

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

- Terminal line controllers worksheet
- RS-232/422 ports on the computer unit worksheet
- VAC/16 controller worksheet
- VDA host adapter worksheet

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

Preceding each worksheet is a sample worksheet that has been filled out for an AViiON 5000 computer with three terminal line

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

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

Figures 10–1 and 10–2 show a sample terminal line controllers worksheet and a copy for you to complete.

Figures 10–3 and 10–4 show a sample RS-232/422 ports on computer unit worksheet and a copy for you to complete.

Figures 10–5 and 10–6 show a sample VAC/16 controller worksheet and a copy for you to complete.

Figures 10–7 and 10–8 show a sample VDA host adapter worksheet and a copy for you to complete.

Sample Worksheet

Terminal Line Controllers Worksheet

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

Figure 10–1 Sample worksheet for terminal line controllers

Terminal Line Controllers Worksheet

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

Figure 10–2 Worksheet for terminal line controllers

Sample Worksheet

Computer Unit RS-232/422 Port Worksheet

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

Figure 10–3 Sample worksheet for RS-232/422 ports on computer unit

Computer Unit RS-232/422 Port Worksheet

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

Figure 10–4 Worksheet for RS-232/422 ports on computer unit

Sample Worksheet

VAC/16 Controller Worksheet

Board no: *0*			Device name: *syac()*				Range of tty lines: *01 –16*
Port No.	**tty Line**	**Device Type**	**Description**	**Port No.**	**tty Line**	**Device Type**	**Description**
0	*01*	*4558 printer*	*officeA1 conn 1100*	**8**	*09*	*D216+*	*officeA10 conn 1118*
1	*02*	*Epson printer*	*officeA2 conn 1102*	**9**	*10*	*VT100*	*officeA11 conn 1120*
2	*03*	*D216+*	*officeA3 conn 110*	**10**	*11*	*D462+*	*officeA12 conn 1122*
3	*04*	*D462+*	*officeA4 conn 1106*	**11**	*12*	*D462+*	*officeA14 conn 1124*
4	*05*	*D216+*	*officeA5 conn 1108*	**12**	*13*	*D462+*	*officeA14 conn 1124*
5	*06*	*D216+*	*officeA6 conn 1110*	**13**	*14*	*D462+*	*officeA18 conn 1128*
6	*07*	*D216+*	*officeA5 conn 1114*	**14**	*15*	*D462+*	*officeA20 conn 1130*
7	*08*	*D216+*	*officeA5 conn 1116*	**15**	*16*	*D462+*	*officeA21 conn 1132*

Figure 10–5 Sample worksheet for a VAC/16 controller

VAC/16 Controller Worksheet

Board no: *0*			**Device name:** *syac()*			**Range of tty lines:** *01 –16*	
Port No.	**tty Line**	**Device Type**	**Description**	**Port No.**	**tty Line**	**Device Type**	**Description**
0				**8**			
1				**9**			
2				**10**			
3				**11**			
4				**12**			
5				**13**			
6				**14**			
7				**15**			

Figure 10–6 Worksheet for a VAC/16 controller

Sample Worksheet

VDA Host Adapter Worksheet

Sheet __1___ of __4___

Board type: *VDA/128* **Board no:** *1* **Device name:** *syac(1)* **Range of tty lines:** *17–271*

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

Figure 10–7 Sample worksheet for a VDA host adapter

VDA Host Adapter Worksheet
Sheet _____ of _____

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

Figure 10–8 Worksheet for a VDA host adapter

Finding the order of controller names in your system file

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

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

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

IMPORTANT: Your system file may have a name other than **system.aviion**. View the appropriate file. Refer to Chapter 15 for information on building kernels and editing system files.

Figure 10–9 shows an excerpt from a system file.

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

Figure 10–9 Automatically configured devices in the system configuration file

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

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

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

Finding the terminal line controller type and cluster controller type

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

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

- VAC/16 or cluster controller port to which a specific terminal is connected.

IMPORTANT: The person installing your computer hardware may have recorded this information on device worksheets supplied in the manual *Setting Up and Installing VMEbus Options in AViiON® Systems*. If these worksheets are available from the hardware installer and they have all the information listed above, then skip this section and continue with "Determining the tty lines for terminal line controller ports" in this chapter. If this information is not readily available, following the instructions in this section to get it.

▶ To determine the terminal line controller type and cluster controller type:

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

```
Current time is 10:21 Tuesday, May 4, 1993. Is this correct (Y/N) [Y]?

Sizing Peripherals....

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

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

VME Async Board 0:

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

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

VME Async Board 1:

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

Press New Line to proceed
```

Figure 10–10 Sample peripherals display from AViiON system diagnostics

On a copy of the terminal line controllers worksheet (Figure 10–2), record each VME async board type that was listed under the "Sizing Peripherals...." line. The names that the diagnostics system uses for the different terminal line controller boards are as follows:

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

For example, using the sample screen in Figure 10–10, you would write "VDA/16" in the Board or Port Type column for Board No. 0, and "VAC/128" in the Board or Port Type column for Board No. 1.

On a copy of the terminal line controllers worksheet (Figure 10–2), record the number of lines for each cluster controller address (Net ID) listed under a VME host adapter. The model numbers for the 8-line and 16-line cluster controllers are as follows:

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

For example, using the sample screen in Figure 10–10, for Board 0, under "Cluster Controllers" you would write "8" in the "No. Lines" column for Address 01 and "16" in the "No. Lines" column for Addresses 02 and 08.

When you finish recording this information, press Enter. If more sizing information appears, record that information also. Continue this process of viewing and recording sizing information until the Main Menu appears.

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

VAC/16

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

VDA host adapter

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

You now have enough information to determine which tty line your DG/UX system assigns to a specific port on any of the terminal line controllers in your computer. In addition, if you know what type of

terminal is connected to each terminal line controller port, you can add these devices to your DG/UX system without having to complete the following section. In this case, you should record the device type for each port on the appropriate tty worksheet.

Next, exit to the SCM by selecting option 4 on the Main Menu, and continue with "Determining the tty lines for terminal line controller ports" in this chapter. If you do not know which type of terminal is connected to each terminal line controller port, continue with "Determining the port where each device is connected."

Determining the port where each device is connected

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

Figure 10–11 shows the System Diagnostics' main menu that appears on the system console.

```
System Diagnostics
         Revision: xx.xx

         Data General Corporation
         Proprietary Use Only

                   Main Menu

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

         Enter choice [1]:
```

Figure 10–11 System Diagnostics' main menu

1. At the main menu, select **2** for the View Tools Menu and press Enter.

   The Tools menu shown below appears.

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

    Enter choice [9]:
```

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

   The Terminal Test Menu shown below appears.

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

    Enter choice [9]:
```

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

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

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

4. Press Enter.

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

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

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

5. Look at the message displayed on each device for which you do not know the port number, and determine the board number, cluster address, and port number for the device. On a copy of the appropriate device worksheet (Figure 10–6 or 10–8) under the specified board number, cluster address (if applicable), and port number, record the type of device (for example, D460 terminal or Model 6640 parallel laser printer) displaying the message. Also record a description that locates the device (for example, office 3B, connector #1356). While the port ID messages are being displayed, the Terminal Test Menu appears on the system console screen.

6. After you finish recording the information for each terminal, select **6** from the Terminal test menu and then press Enter.

   The following prompt is displayed:

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

7. Press Enter to accept the default response to the next prompt. From the Terminal Test Menu, press Enter again to select the Return to Tools menu.

8. From the Tools Menu, press Enter to select Return to Main Menu.

9. At the main menu, exit to the SCM by pressing **4** and Enter.

Determining the tty lines for terminal line controller ports

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

How the DG/UX system allocates tty lines

When you booted your machine, it automatically allocated a specific tty line for each port on each terminal line controller in your computer. Table 10–1 lists the number of tty lines that the DG/UX system allocates to each type of terminal line controller.

Table 10–1 Number of tty lines allocated to terminal line controllers

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

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

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

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

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

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

tty00 to the RS-232/422 port on the duart.
tty01–tty16 to the VAC/16 controller.
tty17—tty272 to the VDA/128 host adapter.

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

tty00–tty15 to the VAC/16 controller.
tty16–tty271 to the VDA/128 host adapter.
tty272 to the RS-232/422 port on the duart.

The DG/UX system assigns each of the 16 tty lines that it allocates to a VAC/16 host adapter to its 16 ports in sequential order. In other words, in the example above where **tty01** through **tty16** are

allocated to the VAC/16 controller, **tty01** is assigned to port 0, **tty02** is assigned to port 1, **tty03** is assigned to port 2, and so on.

Since devices connect to a VDA host adapter through ports on cluster controllers, the DG/UX system allocates specific tty lines to those ports. Table 10–2 shows how the DG/UX system allocates 16 tty lines to each cluster controller address, 01 through 10 hexadecimal (16 decimal).

Table 10–2 tty lines allocated to cluster controller addresses

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

This means that the DG/UX system allocates 16 tty lines to an 8-line cluster controller with one of these addresses. Since an 8-line cluster controller has eight asynchronous ports (ports 0 through 7) and one parallel printer port (port 8), the last seven tty lines allocated to this controller's address are unused. The DG/UX system assigns the 16 tty lines that it allocates to a cluster controller address as follows: the tty line with the lowest number (call it *n*) to port 0; the next tty line with the next highest number (*n*+1) to port 1; the one with the next highest number (*n*+2) to port 2, and so on.

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

box, this tty line is one of the seven unused tty lines allocated to the controller.

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

- **8-line cluster controller (address 01):**

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

- **16-line cluster controller (address 02):**

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

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

1. On your completed copy of the terminal line controller worksheet, find the device name with Configuration File Position 1, then get the tty worksheet that has that device name written on it.
2. Determine the tty line or range of tty lines assigned to this device name using the procedure below. Note that the AViiON 4600 series machines have three asynchronous ports, which could be **tty00**, **tty01**, or **tty02**.

   If the device name is **duart()** or **duart(1)** write "00" in the tty Line column on the tty worksheet for that device name. If the device name is **syac( )**, **syac(1)**, **syac(2)**, **syac(3)**, or **syac(4)**, use the formula in Table 10–3 for the board type of that device name to calculate the range of tty lines assigned to that board (where n = 00), and record this range on the tty worksheet for the board.
3. Using your completed copy of the terminal line controller worksheet, find the device name with previous higher Configuration File Position, and get the tty worksheet that has that device name written on it.

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

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

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

Table 10–3 Lines allocated to systech terminal controllers and cluster controllers

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

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

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

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

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

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

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

Getting information about terminals

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

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

Linesets

The *lineset* establishes the speed and other line characteristics of the port to which the terminal or the modem for a terminal is connected. The port line characteristics and the corresponding terminal or modem line characteristics must match.

Table 10–12 lists the lineset names for terminals that the DG/UX system supports, together with their line characteristics. The DG/UX system uses **9600** as the default lineset name. For information about setting the line characteristics of terminals and modems, see the documentation for the specific devices.

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

Figure 10–12 Lineset names for terminals

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

TERM variables

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

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

For a description of the TERM variables, see the **term** man page.

Listing currently supported TERM variables

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

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

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

Displaying values defined by a TERM variable

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

infocmp *TERM-variable*

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

Determining a terminal's lineset and TERM variable

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

Determining the tty line assigned to the terminal line controller port

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

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

Filling out the tty lines worksheet

To help you keep track of the lineset names and TERM variables for terminals and their associated tty lines, record the values on a copy of Figure 10–14, the tty lines worksheet. Figure 10–13 shows a completed sample of this worksheet.

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

- If a device worksheet lists a terminal for a tty line, then, on the tty lines worksheet, record the tty line, the device type for the terminal, the lineset for the terminal or the modem it uses, and the TERM variable for the terminal.
- If a device worksheet lists a printer for the tty line, then, on the tty lines worksheet, record the tty line and the device type for the printer.
- If a device worksheet does not list a device for a tty line, then, on the tty lines worksheet, record the tty line and write UNUSED beside it.

Sample tty Lines Worksheet — Sheet ___ of ___

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

Figure 10–13 Sample tty lines worksheet

tty Lines Worksheet — Sheet ___ of ___

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

Figure 10–14 Worksheet for tty lines

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

Terminal and port operations

The Port menu provides all the **sysadm** operations necessary for managing terminals and ports. The Terminal menu provides operations for setting up user terminal lines, sometimes called TTYs. The operations in the Terminal menu are streamlined versions of the operations in the Port Services menu, provided to make it easier for you to manage user terminal lines.

The Port Monitor menu operations let you set up and maintain port monitors, which are daemons responsible for attaching services to specific ports. The most common function of a port monitor, for example, is to provide login services for user terminals.

The Port Service menu operations let you set up and maintain assignments of port monitors to specific ports. You provide login capabilities to user terminals, for example, by attaching a port monitor to a particular terminal line. The operations in this menu are generalized for setting up all kinds of port services, not just terminal lines.

The following sections discuss the menus and operations in more detail.

Adding and modifying terminals

The first time you use the `Device-> Port-> Terminal-> Add` operation to add a terminal, the operation checks to see if a **ttymon** port monitor already exists. If none exists, the operation adds and starts a **ttymon** port monitor. The port monitor, named **ttymon1**, is configured to manage user terminals and provide normal login services. All you have to do is add the terminals with the Add operation.

You should not add a port service on a line that is not connected to a terminal or modem. Lines connected to devices such as printers, the asynchronous port on an Uninterruptible Power Supply unit, or ports used for **mterm**(1) connections may produce noise on the line. Unterminated lines also transmit noise back to the system. Noise on the line can cause the port monitor to consume inordinate amounts of CPU time. Noise on the line can also produce strange errors in port monitor log files.

The number of terminals that can be supported by a port monitor is seven less than the value of **HDESLIM**; for example, 1024 – 7 =

1017. In general, however, we recommend that you configure significantly fewer terminals per port monitor.

Additional port monitors can help to share the load incurred by users logging in and logging out. Having multiple port monitors also allows some flexibility in managing user terminals because you can enable and disable a port monitor's terminals as a single unit.

Generally, we recommend that you configure port monitors so that you can readily identify devices and terminal lines associated with the port monitor. For example, you could define a port monitor per cluster box. To add a port monitor, use the operation `Device-> Port-> Port Monitor-> Add`.

The **sysadm** operations Terminal–> Add and Terminal–> Modify present a number of queries. For the Add operation, the defaults are the recommended values for a typical login terminal configuration. For the Modify operation, the defaults are the existing values for the terminal that you are modifying. The prompts for the queries are:

`Controlling port monitor for terminal`
: This prompt appears only if you have more than one **ttymon** port monitor on the system. You need to supply the name (tag) of the port monitor to which the terminal is assigned. The DG/UX system initially has one port monitor, **ttymon1**, for monitoring terminals. This monitor is the default in this query. If you have several **ttymon** port monitors on your system, the operation prompts you for the name of the port monitor to manage the terminal line.

`Tty device(s)`
: Enter the names of the terminal (**tty**) devices from the **/dev** directory; for example, **tty06**, **tty13**.

`TTY definition label`
: Select a terminal definition label. The labels are from the file **/etc/ttydefs**. The number in the label represents line speed. The **M** prefix denotes labels for use with modems, and the **EP** suffix denotes lines with even parity.

`TERM variable`
: Enter the **TERM** variable that will be in effect when the user first logs into the system. The **TERM** variable value that you select indicates the kind of terminal on the line. The **TERM** variable must correspond to an existing **terminfo**(4) database entry. For a complete list of **terminfo** entries, issue this command line at the shell prompt:

```
# ls -CR /usr/share/lib/terminfo/* | more ↵
```

`Disabled response message`

The disabled response message is a text string that the system transmits to the terminal when you disable the port. To represent New Line characters and Tab characters in the message, use **\n** and **\t**, respectively.

`Initial state`

The initial state may be either ENABLED or DISABLED. Users may log in on an enabled terminal. Users cannot log in on a disabled terminal. Users can identify a disabled terminal because the disabled response message, if defined, appears on it.

Adding a single terminal

▶ To add a single terminal to a new or existing hardware configuration:

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

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

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

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

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

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

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

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

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

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

For example, to select the default:

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

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

   Your terminal hardware documentation should specify the correct terminal type. For valid Data General terminal TERM variable settings, see "Using Data General terminals on the DG/UX system." For example, you might specify vt100:

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

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

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

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

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

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

   ```
   OK to perform operation? [yes] ↵
   Terminal /dev/tty07 have been added.
   ```

   If this is the first terminal added to your system, you will also see messages about the default port monitor **ttymon1**. A port monitor configures and controls access to terminals. If no port monitor exists when a terminal is added, the system assigns the default port monitor named **ttymon1** to manage the terminal. (The **ttymon** port monitor replaces the **getty** and **uugetty** programs from pre-5.4 releases of the DG/UX system.)

   You have added a terminal. If you have another terminal to add, select Add and repeat steps 2 though 8 in this section. If you have a

group of identical terminals to add, continue with "Adding a group of identical terminals."

Adding a group of identical terminals

You can add a group of terminals having the same tty definition label and terminal type at one time. The label comes from the **/etc/ttydefs** file, which establishes initial terminal I/O settings. The labels appear as the first field in each line of the **ttydefs** file. For example, you could add ten terminals having an asynchronous terminal label of **9600** and a **vt100** terminal type.

If you have more than 128 terminals in your configuration, you may want to divide them among several port monitors. For how to create multiple port monitors to which you can add the desired groups, see "Creating a port monitor to manage groups of terminals."

If each terminal in a group has a different definition label and terminal type, you must add the terminals one at a time. For how to add terminals one at a time, see "Adding a single terminal."

▶ To add a group of identical terminals to a new or existing hardware configuration:

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

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

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

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

   A tty device number is assigned automatically to each physically attached terminal when the system is booted.

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

   ```
   Tty device(s): ? ↵
     1 tty00
     2 tty01
     3 tty02
     ...
     10 tty09
     11 tty10
   ```

3. Check the display against the planning worksheets that you completed and decide on the terminals you want to add.

4. Select a range of tty devices by specifying a range of menu entries, separated by commas or, for a range, dashes. For example, to specify menu entries 1 through 3 and 11 (terminals tty00 through tty02 and tty10):

```
Tty device(s): 1-3,11 ↵
TTY Definition Label: [9600]
```

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

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

For example, to select the default:

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

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

   Your terminal hardware documentation should specify the correct terminal type. For valid Data General terminal TERM variable settings, see "Using Data General terminals on the DG/UX system." For example, you might specify vt100:

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

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

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

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

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

9. Review your answers and, if they are correct, press Enter. If not, enter no and re-specify any wrong answers. For example:

```
OK to perform operation? [yes] ↵
Terminal /dev/tty00 has been added.
Terminal /dev/tty01 has been added.
Terminal /dev/tty02 has been added.
Terminal /dev/tty10 has been added.
```

If this is the first terminal group added to your system, you will also see messages about the default port monitor **ttymon1**. A port monitor configures and controls access to terminals. If no port monitor exists when a terminal is added, the system assigns the default port monitor named **ttymon1** to manage the terminal.

If more than one **ttymon** port monitor exists on the system, you are asked for the name of the desired port monitor. You can create a tty port monitor as explained in the next section.

You have added a group of terminals. If you have an individual terminal to add, see "Adding a single terminal."

Checking the terminal additions

After you finish adding terminals to DG/UX, you can try them by bringing the environment down to single user status (**init s**) and then back to init 1 or higher.

To ensure that the correct terminal line controller names are listed in the system file, which is the basis for the kernel, see Chapter 15. To select custom local character sets for your terminals, see "Using Data General terminals on the DG/UX system."

Deleting terminals

When you invoke the Delete operation, you may perform the operation on all terminals or only on terminals that you specify. Deleting a terminal will not terminate a login session currently using the terminal line. When the current user logs off, however, no one will be able to log in over the line.

Listing terminals

Select the List operation to display attributes of all terminals on your system. The attributes are those set when you added the terminal. See the section "Adding and modifying terminals" for a discussion of these attributes.

▶ For a current list of terminals and selected variable settings, follow this path through **sysadm**:

```
Device-> Port-> Terminal-> List
```

A typical list follows.

```
/dev/tty00
        port monitor: ttymon1
        ttydefs label: 9600
               state: ENABLED
        TERM variable: vt100
        disabled msg: This terminal is disabled.
        comment: Tty00
```

Enabling and disabling terminals

Use the Enable operation to allow users to log in using a terminal. The port monitor must also be enabled to allow logging into the system. When you enable a terminal, the system sends the login prompt to the terminal.

Use the Disable operation to disallow users from logging in over the terminal. Disabling a terminal does not terminate any login session currently using the terminal. If you defined a disabled-response message for the terminal, the system sends it to the terminal when you disable it.

Using Data General terminals on the DG/UX system

This section presents guidelines for making Data General terminals operational in a DG/UX environment. It covers the following topics:

- Recommended terminal settings
- Selecting character set and emulation mode
- Setting the line discipline

The capabilities of your terminals are based on a number of factors, some of which you can control. Given your terminal and keyboard type, you can set the TERM environment variable using the DG/UX shell and select a terminal hardware emulation mode through the terminal's firmware setup menu to support a specific terminal behavior and desired character set. Definitions of the TERM variable, terminal emulation mode, and character set follow.

TERM variable

The operating behavior of your terminal is determined by associated characteristics located in a compiled database for terminal and printer device capabilities **/usr/lib/terminfo/?/*** where ? stands for the first character of the name, and * stands for the device name. For example, **/usr/lib/terminfo/d/d215** is the

terminfo entry for Data General's DASHER D215 terminal and terminals that behave like it. You select the appropriate terminal behavior by assigning the **terminfo** entry to the TERM variable.

Terminal emulation mode

A terminal emulator is a program that simulates the operation of a particular terminal model. An emulation mode can be defined by a formal standard or by a defacto industry standard (that is, a specific implementation of a terminal). An example of the former is the American National Standards Institute (ANSI) X3.64-1979 document, implemented as the ANSI mode of operation on older Data General terminals. An example of the latter is the command set of the Digital Equipment Corporation VT100 terminal, implemented as the VT100 emulation mode on modern DG terminals. For example, if you have a DASHER D215 terminal that you set to VT100 mode, its keys will be redefined to behave as the keys on a DEC VT100 terminal.

Supported emulation modes follow:

- VT: a set of terminal behaviors built into VT100 terminal series, which are considered Data General's modern terminals.
- DG-UNIX: a Data General proprietary mode designed for UNIX compatibility. It is a superset of the DG mode (terminal behaviors built into older Data General terminals that run on Data General proprietary systems). The DG/UX system does not support DG mode.
- ANSI: a set of terminal behaviors built into Data General's older terminals such as the D211.

Character set

Terminals must be encoded with a character set to manipulate data. A variety of standards is available; your terminals must conform to one. They follow:

- ISO 8859-1 (Western European language conformance).
- VT Multinational (International conformance designed for the Data General VT terminal series); subset of ISO 8859-1.
- DGI (proprietary Data General International).
- U.S. ASCII; a subset of ISO 8859-1, VT Multinational, and DGI.

Your terminals are probably already set up to behave according to one of these standards. However, you can set the desired character set for some terminals.

Recommended terminal settings

If possible, set terminals to operate in VT or ANSI emulation mode in a DG/UX environment. These modes are most compatible with the features a UNIX system expects a terminal to offer.

The following tables detail configurations of Data General terminals, keyboards, emulation modes, and TERM variable settings. For each configuration, it specifies the supported character set and function keys. The legend following the table defines abbreviated terms.

Table 10–4 focuses on the "older" terminals.

Table 10–4 Older terminal configuration information

Terminal Type	7–Bit TERM Variable	7–Bit Character Set	8–Bit TERM Variable	8–Bit Character Set
D210	d210	USASCII	n/a	DGI
D211	d211–7b	USASCII	d211	DGI
D214	d214	USASCII	n/a	DGI
D215	d215–7b	USASCII	d215	DGI
D220	d220–7b	USASCII	d220	DGI
D410	d410–7b	USASCII	d410	DGI
D411	d411–7b	USASCII	d411	DGI
D460	d460–7b	USASCII	d461	DGI
D461	d461–7b	USASCII	d461	DGI
D470C	d470c–7b	USASCII	d470c	DGI
D555	d555–7b	USASCII	d555	DGI
D577	d577–7b	USASCII	d577	DGI
D578	d578–7b	USASCII	d578	DGI

Your TERM variable selection will depend on whether your terminal is configured for 7-bit or 8-bit communications. Each of these terminal types is used with a 6246 keyboard type and runs in ANSI emulation mode. Consult your terminal's hardware documentation for this information.

All function keys used with and without the Shift, Ctrl, and Ctrl-Shift control keys will work properly.

Table 10–5 focuses on the "newer" terminals.

Table 10–5 Newer terminal configuration information

Configuration Elements				Resulting Features	
Terminal Type	**Keyboard Type**	**Emulation Mode**	**TERM Variable**	**Internat'l Character Set**	**Function Keys Supported**
D1400i	6488	wyse 60	d1400i	None	F1–F9
	6488	wyse 60	wyse60	None	F1–F9
	6488	wyse 50+	wyse50	None	F1–F9
	6488	vt100	vt100	VTM	F1–F4, Num Keypad (,456789)
	6488	vt220–7	vt220	VTM	F1–F12
	6488	vt220–8	vt220	VTM	F1–F12
D216 D216E	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6348	vt100	d216	None	C1–C4
D216+ D216E+	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6488	vt100	vt100	None	F1–F4, Num Keypad (,456789)
	6348	vt100	d216	None	C1–C4
	6488	vt100	d216	None	F1–F4
	6488	vt100	d216	DGI	F1–F4
	6348	dg–unix	d216+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D217	6348	vt100	vt100–fk	None	C1–C4, F1–F12
	6488	vt100	vt100–fk	None	F1–F12
	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6488	vt100	vt100	None	F1–F4, Num Keypad (,456789)
	6348	vt100	d217	None	C1–C4, Num Keypad (,456789)
	6488	vt100	d217	None	F1–F4, Num Keypad (,456789)
	6348	dg–unix	d217–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d217–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift

Continued

Configuration Elements				Resulting Features	
Terminal Type	**Keyboard Type**	**Emulation Mode**	**TERM Variable**	**Internat'l Character Set**	**Function Keys Supported**
D230C	6384	ANSI	d220–7b	None	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	ANSI	d220	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	ANSI	d230c	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D412	6348	vt220	vt220	VTM	F1–F18
	6348	vt220	d412	VTM	F1–F15
D412+	*6348	vt320	vt220	VTM/ISO	F1–F18
	*6488	vt320	vt220	VTM/ISO	F1–F12
	*6348	vt320	vt412	VTM/ISO	F1–F15
	6348	dg–unix	412+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d412+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D413	6348	vt320	vt220	ISO	F1–F18
	6488	vt320	vt220	ISO	F1–F12
	6348	dg–unix	d413–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d413–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D462	6348	vt220	vt220	VTM	F1–F18
	6348	vt220	d462	VTM	F1–F15
D462+	*6348	vt320	vt220	VTM/ISO	F1–F18
	*6488	vt320	vt220	VTM/ISO	F1–F15
	*6348	vt320	d462	VTM/ISO	F1–F15
	6348	dg–unix	d462+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d462+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D463	6348	vt320	vt220	ISO	F1–F18
	6488	vt320	vt220	ISO	F1–F12
	6348	vt320	d463	ISO	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	dg–unix	d463–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d463–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift

LEGEND

None	No international character set support is provided; however, the U.S. ASCII character set is supported.
6348	CEO–style 107–key keyboard.
6246	Older version of CEO–style 107–key keyboard.
6488	101–key keyboard (PC/AT layout; 102 keys for international.

DGI	Proprietary DG International.
VTM	VT Multinational.
ISO	ISO 8859–1 (requires setup sequence on D412+ and D462+).
*	For D412+ or D462+ terminals, must be a firmware Revision 3 or higher to support correctly in vt320 mode.
Num Keyboard (,456789)	Includes these keys only: , 4 5 6 7 8 9
Shift, Ctrl, Ctrl–Shift	Keys used in conjunction with (press and hold while pressing) a function key F1–F15.
F1–F4	Function keys F1, F2, F3, and F4 only.
vt100/vt220/vt320 emulation	DEC VT100, VT220, and VT320 terminal emulation will cause a mismatch between a keyboard's key labels and their corresponding functions. Use the adhesive labels provided with your hardware documentation for correct key representation. Only the function keys operate correctly in VT100, VT220, or VT320 emulation modes. The Shift and Ctrl keys are not supported. One exception, however, is the D1400i terminal which can operate in any of these modes and correctly matches function key labels to corresponding functions.
C1–C4	Four keys labeled C1, C2, C3, and C4 on the keypad.

Selecting character set and emulation mode

You can explicitly select your terminal's character set for only the D217, D413, and D463 terminal models. Most older terminals are set to only one character set. See your hardware documentation for more information.

Select the desired terminal emulation mode by invoking the terminal's configuration menu or by setting the DIP switches on the rear of the terminal, depending on the model of terminal.

Using the D216 as an example, press Shift-N/C to invoke the terminal's port menu so that you can select the VT100 operating mode.

Setting the line discipline

Line discipline is the term that applies to the options that interpret terminal characteristics.

Terminal options that are probably most important to you are the definitions of the line-editing and control keys.

Table 10–6 lists the most frequently used keys with their common settings.

Table 10–6 Commonly used line-editing and control keys

Function	Definition	Common Setting
intr	Terminates a process.	Ctrl–C
quit	Terminates a process but dumps an image of memory that can be examined later.	Ctrl–\
erase	Removes a character and closes up the space.	Ctrl–H
kill	Deletes a line.	Ctrl–U
eof	End–of–file (logs you out); also terminates interactive input for commands such as **mailx** (an electronic mail facility that enables the exchange of messages) and **cat** (a command that allows you to send text to a file via the terminal).	Ctrl–D
start	Resumes scrolling of text on the screen.	Ctrl–Q
stop	Stops scrolling of text on the screen.	Ctrl–S
susp	Suspends temporarily an active job.	Ctrl–Z
werase	Removes a word and closes up the space.	Ctrl–W

With the **stty** (set terminal type) command, you can find out the current settings for the line discipline. Typical output for the **stty everything** shell command follows:

```
% stty everything
speed 9600 baud; line = 1;
rows = 50; columns = 80; ypixels = 754; xpixels = 739;
intr = ^c; quit = ^|; erase = ^h; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -parext
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
  ixon -ixany -ixoff -imaxbel
isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop echoctl -echoprt echoke -defecho -flusho -pendin iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
%
```

The caret (^) represents the Control key.

If you have an older Data General terminal operating in ANSI emulation mode, you **must** change the line discipline to enable tab expansion and disable new line expansion to the carriage return/new line function. Type this command:

```
% stty nl tab3 ↵
```

To change any key settings, use this command format:

stty *key-name character*

When you redefine keys, use the caret symbol (^) to represent the Control key. Also, you must precede it (and any other special character that is used as a metacharacter by the shell), with a backslash (\) or you must surround the entire key definition with a pair of single or double closed quotation marks. (' ' or " "). A metacharacter is a symbol that stands for something other than itself, such as the caret (^) which in the shell represents the first character in the line. Examples follow:

```
% stty intr \^C ↵
% stty erase \^\? ↵
% stty eof \^D ↵
```

Type the **stty** command again to see the change in the line discipline.

When selecting a new key definition, make sure that the key is not already being used. Also, if you use the **editread** facility, those keys should not conflict with keys set through the **stty** command. Refer to *Using the DG/UX™ System Editors* for detailed information on **editread**.

Once you make changes to your line discipline from the shell, they will remain in effect only while you are currently logged in. You will have to reset them each time you log in. If you prefer, you can put them in a setup file (such as your **.login**) so that they will be set automatically each time you log in.

See the **stty**(1) manual page for more information on setting line discipline.

Setting the TERM variable

Set the **TERM** environment variable. Using the D216 terminal in a C shell environment as an example, you would use the following shell commands:

```
% setenv TERM vt100 ↵
% tput init ↵
```

The **tput** command sets up such terminal capabilities as cursor movement, function key sequences, and screen highlighting (reverse video).

IMPORTANT: If you are having problems with terminal control, check to make sure that the **TERM** variable is set correctly for your terminal. Also try executing the command **tput reset** to reset your terminal to its normal operating mode.

Managing port monitors

The DG/UX system provides two types of port monitors. The first, **ttymon**, controls access to the system over specified ports, starting programs to provide services such as **login** service to users or applications who attempt to access the system over those ports. The second monitor, **listen**, monitors ports used for TLI (Transport Layer Interface) services. For more information, see *Programming with TCP/IP on the DG/UX™ System*.

The Port Monitor menu provides operations for adding, deleting, modifying, and listing port monitors. It also provides operations for enabling, disabling, starting, and stopping them. By default, SAF starts monitors when it comes up to run level 2, 3, or 4. To change this behavior, edit the second field of the `saf` entry in **/etc/inittab**. See Chapter 3 for the format of **inittab**(4) entries.

Adding and modifying port monitors

To add or modify a port monitor, you need to understand the attributes that define the port monitor. These attributes, discussed below, correspond to the queries in the Add and Modify operations.

`Port Monitor Type`
: The port monitor type may be either **ttymon**, for providing services to terminal users and UUCP callers, or **listen**, for monitoring ports used for TLI communication. If you have written your own port monitor, specify the type here.

`Port Monitor Tag`
: The port monitor tag is a unique name that distinguishes this monitor from others on your system. You may want the name to indicate the ports or terminal lines monitored.

`Command to Start Port Monitor`
: This is the command line that the system should use to start the monitor. Unless you know of options you wish to specify, take the default. See the **ttymon**(1M) manual page or the **listen**(1M) manual for more information on command line invocation.

`Version Number`
: This is the version number of the port monitor. The version number defines the format of the port monitor administrative file.

`Initial Run State`
: The initial run state determines whether or not the system will start the port monitor at boot time. You may choose either STARTED or STOPPED. If you select STOPPED, the

port monitor will not start until you start it explicitly with the Start operation or the **sacadm**(1M) or **admportmonitor**(1M) commands.

`Start State`
: The start state determines whether or not the port monitor will accept requests for services when it starts. You may choose either ENABLED or DISABLED. If enabled, the port monitor monitors its ports and accepts connection requests. If disabled, the monitor monitors its ports but does not accept connection requests.

`Restart Count`
: The restart count determines how many times the Service Access Controller (SAC) will attempt to restart the port monitor if it fails. If the port monitor will not start after this number of retries, SAC places it in a FAILED state.

`File Name of Configuration Script`
: A configuration script can initialize the port monitor by pushing STREAMS modules onto the port monitor's stack or by setting environment variables for the port monitor. When the monitor starts, it executes the commands in this script. The configuration script is optional. For more information on configuration scripts, see the **doconfig**(3N) manual page.

`Comment`
: This comment may be any text that you wish to add to describe the port monitor for your own records.

Adding a port monitor to manage groups of terminals

Port monitors are used to configure and control access to groups of terminals. The number of terminals you attach to a port monitor depends on your particular hardware configuration and your performance requirements. The default port monitor **ttymon1** is created automatically when the first terminal is added. If you have more than 128 terminals in your configuration, you may want to divide them among several port monitors.

▶ To add a port monitor to a new or existing hardware configuration:

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

   ```
   Device-> Port-> Port Monitor-> Add
   ```

   This operation creates a directory structure for the new port monitor and adds an entry for it in the Service Access Controller's database. You assign a port service to each terminal that is controlled by the port monitor.

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

```
Port monitor type: [ttymon]
```

2. You may select from two types of port monitors: **ttymon** or **listen**. If you are adding a port monitor for typical asynchronous tty lines, press Enter to accept the default **ttymon**. For a Transport Layer Interface (TLI)-based network system, enter **listen**. For example:

```
Port monitor type: [ttymon] ↵
Port monitor tag:
```

3. Provide a tag (name) name of up to 14 alphanumeric characters for the port monitor. For example,

```
Port monitor tag: ttymon2 ↵
Command to start port monitor: [/usr/lib/saf/ttymon]
```

4. The prompt sets the command to start the port monitor, and the version of the port monitor. You can usually accept the default answers of **/usr/lib/saf/ttymon** and release **1** although your port monitor name differs from the displayed program name, ttymon. For example:

```
Command to start port monitor: [/usr/lib/saf/ttymon] ↵
Version number: [1] ↵
Initial run state: [STARTED]
```

5. Select the initial run state for the port monitor: STARTED or STOPPED. The default choice, STARTED, means that the Service Access Controller starts the port monitor immediately each time the Service Access Controller starts at boot time. If you select STOPPED, the port monitor does not become active until you start it using **sysadm**, or the **sacadm** or **admportmonitor** commands. We recommend the default, STARTED. For example:

```
Initial run state: [STARTED] ↵
Start state: [ENABLED]
```

6. Specify the state of the port monitor when started: ENABLED or DISABLED. If you select ENABLED (the default), the port monitor begins management duties and accepts connection requests when started. If you select DISABLED, the port monitor runs but does not accept connection requests. For example:

```
Start state: [ENABLED] ↵
Restart count: (0-10) [0]
```

7. Specify the restart count, which is the number of times the Service Access Controller should attempt to restart the port monitor if it fails. If the port monitor cannot be restarted after this number of tries, it is placed in the FAILED state. For example:

```
Restart count: (0-10) [0] 5↵
File name of configuration script:
```

8. If there as a file that contains the port monitor configuration script, specify its full pathname here. When the port monitor is added, the contents of this file are copied to the port monitor configuration script file. The port monitor reads configuration files only when it starts. Changes to a script file made while the monitor is running are not effective until you restart the monitor. For more on configuration scripts, see the **doconfig**(3N) manual page.

   If you don't want to specify a configuration script, press Enter.

```
File name of configuration script: ↵
Comment:
```

9. You may supply a comment about the port monitor to be displayed when you select an operation that lists information about the port monitor. Or you can press Enter to skip the comment. For example:

```
Comment: Port monitor to control ttys on hallway12. ↵
OK to perform operation: [yes]
```

10. Review your answers and, if they are correct, press Enter. If not, enter no and re-specify any wrong answers. For example:

```
OK to perform operation: [yes] ↵
Port monitor ttymon2 has been added.
```

    The final message informs you that the named port monitor **ttymon2** was successfully added.

Deleting port monitors

The Delete operation removes port monitors. You may specify one or more existing port monitors by tag, or you may perform the operation on all monitors.

Deleting a port monitor terminates all sessions that it started.

Listing port monitors

The List operation displays information on all port monitors.

▶ For a current list of port monitors and selected variable settings, follow this path through **sysadm**:

```
Device-> Port-> Port Monitor-> List
```

An example listing follows:

```
PMTAG     PMTYPE    FLGS RCNT STATUS    COMMAND
tcp       listen    -    3    ENABLED   /usr/lib/saf/listen tcp #listener for tcp
ttymon1   ttymon    -    0    ENABLED   /usr/lib/saf/ttymon     #
```

The # at the end of each line is a comment delimiter.

Enabling and disabling port monitors

You may perform the Enable or Disable operations any time a port monitor is running. By default, the system starts and enables port monitors when the system comes up to run levels 2, 3, and 4.

An enabled port monitor monitors its assigned ports and accepts requests for login connections to the system. A disabled port monitor does not accept requests at any of its ports. Disabling a port monitor terminates all sessions that it started.

Starting and stopping port monitors

If you have not defined your port monitors to start at boot, you need to start them explicitly either with the Start operation or with the **sacadm**(1M) or **admportmonitor**(1M) commands.

Performing the Stop operation for a port monitor does not terminate any connections already established by the port monitor.

Managing port services

After you have added port monitors on your system, you need to assign port services. A port service associates a port (such as a TTY line) with an existing port monitor, describing the nature of the service that the port requires. A port service definition consists of attributes such as the pathname of the device to monitor, the program to run when a user connects on the line, settings determining how communication should occur over the line, and so on. You assign a port service for every port that you want a monitor to manage.

You should not add a port service on a line that is not connected to a terminal or modem. Lines connected to devices such as printers, the

asynchronous port on an Uninterruptible Power Supply unit, or ports used for **mterm**(1) connections may produce noise on the line. Unterminated lines also transmit noise back to the system. Noise on the line can cause the port monitor to consume inordinate amounts of CPU time. Noise on the line can also produce strange errors in port monitor log files. Noise on the line can also produce strange errors in the port monitor log files.

Adding and modifying port services

Before invoking the Add or Modify operations, familiarize yourself with the queries that they present. Some queries apply to both **ttymon** and to **listen** port services while others apply only to one or the other.

`Controlling port monitor for service`
: This is the tag (name) of the port monitor that will monitor the port. You should have already added this port monitor with the **sysadm** operation `Device-> Port-> Port Monitor-> Add`.

`Port service tag`
: This is a name representing this service definition. This tag may be any name that you choose. The name should be unique for the given port monitor. For port services added with the Add operation of the Terminal menu, the tag is the name of the TTY, for example, `tty02`.

`Service Userid`
: Every service runs as a process on your system. The user ID you select determines which user will own the process. The user ID must already exist. Typically, the user ID is **root**.

`Create utmp entry`
: You may choose whether or not the system creates an **/etc/utmp** entry for the connecting user. Several system programs, including **who**, **write**, and **login**, depend on the **utmp** file for user information.

`File name of configuration script`
: You may specify a configuration script to modify the behavior of the port monitor. This configuration script affects only the service on this port. The port monitor copies the script to its own configuration script when it starts. For more information on configuration scripts, see the **doconfig**(3N) manual page.

`Comment`
: This comment may be any text that you choose. You may use the comment field to contain a description of the port service for your records.

`Initial state`

The initial state may be either ENABLED or DISABLED. An enabled line accepts requests for connections to the service. A typical login line should be enabled to allow users to log into the system.

A disabled line refuses requests. When you disable a line, the system transmits a disabled-response message on the line. You enter the disabled-response message in the "Disabled response message" query, below.

`Version number`

This field is the version number of the port monitor.

The following queries appear when you add or modify **ttymon** services:

`Path name of terminal device`

Specify the pathname of the port or terminal. This path should indicate the port's entry in the **/dev** directory, for example, **`/dev/tty04`**.

`TTY definition label`

This label corresponds to an entry in the **/etc/ttydefs** file. The TTY definition determines terminal I/O characteristics for the line, such as line speed and whether or not the line is for a modem. In the TTY definitions provided with the system, the number in the label indicates line speed. The prefix **`M`** indicates that the definition is intended for modems, and the suffix **`EP`** indicates that the line is even parity.

`Service command`

The service command is the shell command line that the monitor should invoke for a calling user when the line is enabled. For a typical login line, the service is **/usr/bin/login**.

`Hangup`

This attribute causes the system to "hang up" the line by setting line speed to zero before initializing the line. This feature is useful if the line is connected to a modem.

`Connect on carrier`

This feature causes the port monitor to invoke the service as soon as it receives a carrier indication. Use this feature only if you are sure of the baud rate of the incoming caller and you know that the incoming caller does not require a prompt to begin the session.

`Bidirectional`

A bi-directional line allows local users and programs to call

out on the line and outside users to call in on the line. For a normal terminal line, you do not want to set the line for bi-directional use. If the line is connected to a modem used for dialing out as well as for receiving calls, you want to set the line for bi-directional use.

`Wait-read value`
: This value determines how many New Line characters the port monitor must receive before it sends the prompt out on the line. If you specify 0, the port monitor waits for any character. If you specify "none," **ttymon** sends a prompt without waiting for characters.

`Timeout`
: This value determines how long the line may be inactive at the login prompt before the port monitor terminates the session. An inactive line is one over which the port monitor detects no transmission of characters. For no time-out period, specify zero seconds.

`Prompt message`
: The port monitor transmits the prompt message when it places a port in the ENABLED state. To represent New Line characters and Tab characters in the message, use `\n` and `\t`, respectively.

`Modules to be pushed`
: Enter the additional STREAMS modules that you want pushed to handle the line. By default, lines connected to a **syac** already have STREAMS modules **ldterm** and **ttcompat** pushed. For information on STREAMS, see *Programming in the DG/UX™ Kernel Environment* and *UNIX System V Release 4: Programmer's Guide: STREAMS*.

`Disabled response message`
: Enter the message that you want the port monitor to transmit when callers attempt to use a disabled line. To represent New Line characters and Tab characters in the message, use **\n** and **\t**, respectively. The default is no message.

The following queries appear when you add or modify **listen** services:

`Service type`
: You may select either of two values for this field:

  `Spawn a service`
  : Select this value to cause the port monitor to invoke the service that you specify in the next field, `Service command or STREAMS pipe`, whenever a connection request is received.

`Pass file descriptor to standing server`
: Select this value to cause the port monitor to pass the file descriptor for this connection through the pipe you specify in the next query, `Service command or STREAMS pipe`, whenever a connection request is received. The pipe is connected to a standing server, a server that is currently running.

`Service command or STREAMS pipe`
: If you selected `Spawn a service` for the service type, enter the service to be started. If you selected `Pass file descriptor to standing server` in the previous query, enter the name of the STREAMS pipe to be used for passing file descriptors.

`Modules to be pushed`
: Enter the names of the STREAMS modules to be pushed. After popping all modules already on the stream, the operation pushes the specified modules in the order in which you enter them.

`Server's private address`
: Enter the address that **listen** should monitor. **listen** will dispatch calls at this address to the designated service. The address must be unique.

Deleting port services

Use the Delete operation to remove port service definitions. You may perform the operation on all port services or only on those specified.

Listing port services

The operation to list port services displays definitions for all port services. To display current port service definitions, use the **sysadm** operation Device–> Port–> Port Service–> List or the command **pmadm –l** discussed in Chapter 11.

A typical list follows:

```
PMTAG           PMTYPE          SVCTAG          FLGS ID       <PMSPECIFIC>
tcp             listen          0               -    root
\x00020ACE00000000000000000000000000 - c - /usr/lib/saf/nlps_server #NLPS
server
tcp             listen          lpd             -    root
\x000202030000000000000000000000000 - p - /var/spool/lp/fifos/listenBSD
#Berkley lp service
tcp             listen          lp              -    root     - - p -
/var/spool/lp/fifos/listenS5 #lp service
tcp             listen          failover        -    root
\x000252D00000000000000000000000000 - c - /usr/bin/failoverd #
tcp             listen          shareddisk      -    root
\x0002539800000000000000000000000000 - c - /usr/sbin/shareddiskd #Shared
disk coordination
```

Enabling and disabling port services

Use the Enable operation to make a port service available. The service is available provided the port monitor is also enabled. For typical terminal lines, this means that users can log into the system.

Use the Disable operation to make a port service unavailable. When the port monitor detects an attempt to access a disabled service, it transmits the disabled-response message defined for the port service.

End of Chapter

11 Service Access Facility (SAF)

The DG/UX system provides port services through the Service Access Facility (SAF). Ports on your system include terminal lines and the lines you use for UUCP connections.

You can manage port services two ways. One way is by using the operations found in the **sysadm** Device–> Port menu. Chapter 10 covers these operations. The second way to manage ports, covered in this chapter, is with SAF commands invoked at the shell level.

SAF provides the functions that underlie the **sysadm** operations you use to manage port services. Its features generalize the procedures for service access so that login access on the local system and network access to local services are managed in essentially similar ways.

Overview of the Service Access Facility

SAF provides a mechanism for uniform access to services. The main components of this generalized access procedure are the commands for installing, configuring, and maintaining port monitors and services and the administrative or database files in which port monitor and service information is stored.

From the point of view of SAF, a service is a process that is started. There are no restrictions on the functions a service may provide.

SAF consists of a controlling process called the Service Access Controller (SAC) and two administrative levels corresponding to two levels in the supporting directory structure. Figure 11–1 shows the SAF process structure.

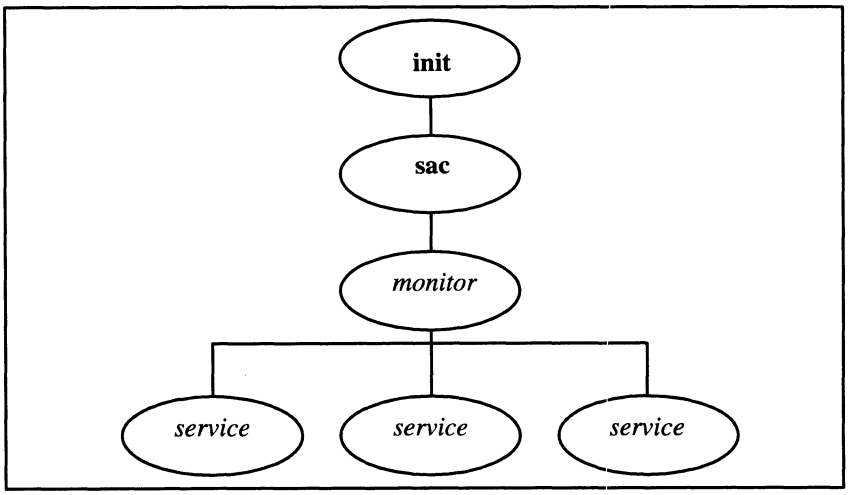


Figure 11–1 SAF process structure

The top administrative level is concerned with port monitor administration, the lower level with service administration. The SAC process, which starts when you take the system to run level 2, 3, or 4, is responsible for starting and maintaining any port monitors that you define. A port monitor is in turn responsible for starting and maintaining any port services that you have defined.

Systems may access services using a variety of port monitors, including port monitors written expressly for a user's application. The section on **listen** describes the administrative tasks required to provide access to services over any network that conforms to the Transport Layer Interface (TLI) protocol. For information on TLI, see *Programming with TCP/IP on the DG/UX™ System*.

From an administrative point of view, SAF consists of the following components, each of which is described in this section:

- The Service Access Controller (SAC)
- A per-system configuration script
- The SAC administrative file
- The SAC administrative command **sacadm** or **admportmonitor**(1M)
- Port monitors
- Optional per-port monitor configuration scripts
- An administrative file for each port monitor
- The administrative command **pmadm** or **admportservice**(1M)
- Optional per-service configuration scripts

The Service Access Controller, the administrative files, and the per-system, per-port monitor, and per-service configuration scripts are described in this section. The administrative command **sacadm** is described under "Port monitor management" and the administrative command **pmadm** is described under "Service management." The **sysadm** utility performs SAC administrative tasks with the **admportmonitor**(1M) command and port monitor-specific administrative tasks with the **admportservice**(1M) command.

The Service Access Controller

The Service Access Controller (SAC) is the overseer of the server system. It is the SAF's controlling process. SAC is started by **init**(1M) by means of an entry in **/etc/inittab**. Its function is to maintain the port monitors on the system in the state you specified. These states include: STARTING, ENABLED, DISABLED, STOPPING, NOT RUNNING, and FAILED. A port monitor enters the FAILED state if SAC cannot start it after a specified number of tries.

The administrative command **sacadm** is used to tell SAC to change the state of a port monitor. **sacadm** can also be used to add or remove a port monitor from SAC supervision and to list information about port monitors known to SAC. The **admportmonitor**(1M) command also provides these services.

SAC's administrative file contains a unique tag for each port monitor known to SAC and the pathname of the command used to start each port monitor.

SAC performs three main functions:

- Customizes its own environment
- Starts the appropriate port monitors
- Polls its port monitors and initiates recovery procedures when necessary

During initialization, SAC customizes its own environment by invoking the per-system configuration script. Next, it reads its administrative file to determine which port monitors are to be started. For each port monitor it starts, it interprets the port monitor's configuration script, if one exists. Finally the port monitors specified in the administrative file (for example, **ttymon**) are started.

Once the port monitors are running, SAC polls them periodically for status information. The **sac**(1M) command line option, **-t**, allows you to control polling frequency. By examining the **saf** entry in the **/etc/inittab** file, you can see that the default is to poll every 45 seconds. When the port monitor gets a request for status from SAC, it must respond with a message containing its current state (for example, ENABLED). If SAC does not receive a response, it assumes the port monitor is not running. If a port monitor that should be running has stopped, SAC assumes it has failed and takes appropriate recovery action.

SAC restarts a failed port monitor if a nonzero restart count was specified for the port monitor when it was created (see the **sacadm** command, described under "Port monitor management" and in the **sacadm**(1M) manual page.

SAC is the administrative point of control for all port monitors (and therefore for all ports on the system). The administrative commands **sacadm**(1M) and **pmadm**(1M) pass requests to SAC, which in turn communicates with the port monitors. These requests include enabling a disabled port monitor so that it begins accepting service requests on its ports; starting port monitors that were previously killed; and listing the current state of all port monitors on the system.

The per-system configuration file

The prototype per-system configuration file, **/etc/saf/_sysconfig.proto**, is delivered empty. You can customize the environment for all services on the system by writing a command script in the interpreted language described in the *Managing TCP/IP on the DG/UX™ System* manual and in the **doconfig**(3N) manual page and placing this command script in the **_sysconfig** file. The per-system configuration script is interpreted by SAC when SAC is started. SAC is started when the system enters multiuser mode.

Per-port monitor configuration scripts

Per-port monitor configuration scripts (**/etc/saf/***pmtag***/_config**) are optional. They allow you to customize the environment for any given port monitor and for the services that are available through the specific collection of access points for which that port monitor is responsible. Per-port monitor configuration scripts are written in the same language used for per-system configuration scripts.

The per-port monitor configuration script is interpreted when the port monitor is started. The port monitor is started by SAC after SAC has itself been started and after it has run its own configuration script, **/etc/saf/_sysconfig**.

The per-port monitor configuration script may override defaults provided by the per-system configuration script.

Per-service configuration scripts

Per-service configuration scripts allow you to customize the environment for a specific service. For example, a service may require special privileges that are not available to the general user. Using the language described in the **doconfig**(3N) manual page, you can write a script that will grant or limit such special privileges to a particular service offered through a particular port monitor.

The per-service configuration may override defaults provided by higher-level configuration scripts. For example, the per-service configuration script may specify a set of STREAMS modules other than the default set.

The SAC administrative file

SAC's administrative file contains information about all the port monitors for which SAC is responsible. The prototype SAC administrative file, **/etc/saf/_sactab.proto**, contains a comment

line containing the SAC version number and an entry for the **tcp** port monitor. You add port monitors to the system by using the administrative command **sacadm** with the **-a** option to make entries in SAC's administrative file. **sacadm** is also used to remove entries from SAC's administrative file. As an alternative, you can use the **admportmonitor**(1M) command or **sysadm**'s `Device-> Port-> Port Monitor` operations to perform the same functions.

IMPORTANT: We recommend that you do not change the SAC administrative file except through the **sacadm**(1M), **admportmonitor**(1M), or **sysadm**(1M) utilities. If you choose to modify the SAC administrative file, do not change it while SAF is running.

Each entry in SAC's administrative file contains the following information:

PMTAG

A unique tag that identifies a particular port monitor. You are responsible for naming a port monitor. This tag is then used by the Service Access Controller (SAC) to identify the port monitor for all administrative purposes.

PMTAG may consist of up to 14 alphanumeric characters.

PMTYPE

The type of the port monitor. In addition to its unique tag, each port monitor has a type designator. The type designator identifies a group of port monitors that are different invocations of the same entity. **ttymon** and **listen** are examples of valid port monitor types. The type designator is used to facilitate the administration of groups of related port monitors. Without a type designator, you have no way of knowing which port monitor tags correspond to port monitors of the same type.

PMTYPE may consist of up to 14 alphanumeric characters.

FLGS The flags that are currently defined are:

d When started, do not enable the port monitor

x Do not start the port monitor.

If no flag is specified, the default action is taken. By default a port monitor is started and enabled.

RCNT The number of times a port monitor may fail before being placed in a failed state. Once a port monitor enters the failed state, SAC will not try to restart it. If a count is not specified when the entry is created, this field is set to **0**. A

restart count of **0** indicates that the port monitor is not to be restarted when it fails.

COMMAND
The command line that starts the port monitor. The first component of the string, the command itself, must be a full pathname.

The example below shows the contents of a sample SAC administrative file as listed by the **sacadm** command.

```
# sacadm -l ↵
PMTAG     PMTYPE    FLGS RCNT STATUS    COMMAND
tcp       listen    -    3    ENABLED   /usr/lib/saf/listen tcp #listener for tcp
ttymon1   ttymon    -    0    ENABLED   /usr/lib/saf/ttymon #
```

The # character at the end of each line is a comment delimiter.

The port monitor administrative file

Each port monitor has its own administrative file, **/etc/saf/***pmtag***/_pmtab**. The **pmadm**(1M) command is used to add, remove, or modify entries in this file. The **admportservice**(1M) command also performs these functions. Each time a change is made, the corresponding port monitor is told to reread its administrative file.

IMPORTANT: We recommend that you do not change a port monitor administrative file except through the **pmadm**(1M) or **admportservice**(1M) commands. If you choose to modify a port monitor administrative file, do not change it while the port monitor is running.

Each entry in a port monitor's administrative file defines how the port monitor should treat a specific port and what service is to be invoked on that port. Some fields must be present for all types of port monitors. Each entry must include a service tag to identify the service uniquely and an identity to be assigned to the service when it is started (for example, **root**). The combination of a service tag and a port monitor tag uniquely define an instance of a service. The same service tag may be used to identify a service under a different port monitor. The record must also contain port monitor specific data such as the prompt string which is meaningful to **ttymon**. In general, each type of port monitor provides a command that takes the necessary port monitor-specific data as arguments and outputs these data in a form suitable for storage in the file. The **ttyadm**(1M) command does this for **ttymon** and **nlsadmin**(1M) does it for **listen**.

Each entry in the port monitor administrative file must contain the following information.

SVCTAG
A unique tag that identifies a service. This tag is unique only for the port monitor through which the service is available. Other port monitors may offer the same or other services with the same tag. A service requires both a port monitor tag and a service tag to identify it uniquely.

SVCTAG may consist of up to 14 alphanumeric characters.

FLGS Flags with the following meanings may currently be included in this field:

x Do not enable this port. By default the port is enabled.

u Create a **utmp** entry for this service. By default no **utmp** entry is created for the service.

Note that port monitors may ignore the **u** flag if creating a **utmp** entry for the service is not appropriate to the manner in which the service is to be invoked. Some services may not start properly unless **utmp** entries have been created for them (for example, **login**).

ID The identity under which the service is to be started. The identity has the form of an existing login name.

PMSPECIFIC
Examples of port monitor-specific information are addresses, the name of a process to execute, or the name of a STREAMS pipe through which to pass a connection. See the **ttyadm**(1M) manual page for information about additional **PMSPECIFIC** items.

COMMENT
A comment associated with the service entry.

Each port monitor administrative file must contain one special comment of the form:

```
# VERSION=value
```

where *value* is an integer that represents the port monitor's version number. The version number defines the format of the port monitor administrative file. This comment line is created automatically when a port monitor is added to the system. It appears on a line by itself, before the service entries.

The example below shows the contents of a sample **ttymon** administrative file as listed by the **pmadm** command.

```
# pmadm -l -p ttymon1 ↵
PMTAG    PMTYPE SVCTAG  FLGS ID   <PMSPECIFIC>
ttymon1 ttymon ttymon1 u    root /dev/tty00 bhr 8 /usr/bin/login 60 \
       M1200 - login:  -  #
```

The # character at the end of each line is a comment delimiter.

Note that everything in the PMSPECIFIC column is specific to a **ttymon** port monitor. The listing for a **listen** administrative file, for example, will contain a different set of entries in this column. Port-monitor specific information is formatted by the port monitor's administrative command, in this case **ttyadm**. The **ttyadm** command is included as part of the **pmadm** command when it is used with the **–a** option. See "Adding a service," under "Service management."

To maintain the integrity of the system, it is strongly recommended that changes in the SAC and port monitor administrative files be made with the **sacadm** and **pmadm** commands, not by editing the files. SAC does not recognize changes in some of the fields in these files unless they are made using the appropriate administrative command. Editing the file directly can lead to unexpected results.

Port monitor management

The SAF administrative model is hierarchical. The highest level is concerned with port monitor administration. The lower level is concerned with service administration and is discussed under "Service management." At the level of port monitor administration, port monitors may be added, removed, started, stopped, enabled, or disabled. Other functions performed at this level include requesting port monitor status information, replacing a per-system configuration file, installing or replacing a per-port monitor configuration file, and requesting that a port monitor read its administrative file.

Configuration scripts are described under "Printing, installing, and replacing per-service configuration scripts." Requesting that a port monitor read its administrative file is under "Reading the administrative files."

The SAC administrative command: sacadm

sacadm is the administrative command for the upper level of the Service Access Facility hierarchy and therefore for port monitor administration (see the manual page **sacadm**(1M). Under SAF, port monitors are administered by using the **sacadm** command to make changes in SAC's administrative file. **sacadm** performs the functions listed below.

- Print requested port monitor information from the SAC administrative file
- Add or remove a port monitor
- Enable or disable a port monitor
- Start or stop a port monitor
- Install or replace a per-system configuration script
- Install or replace a per-port monitor configuration script
- Ask SAC to reread its administrative file

Each function is discussed in one of the following sections.

Printing port monitor status information

Unless you already know the type of a port monitor, you may need to use the most general form of the command (**sacadm -l**) to find out what the valid type and tag names are.

```
sacadm -L [ -p pmtag | -t type ]
sacadm -l [ -p pmtag | -t type ]
```

pmtag is the tag associated with the port monitor that is being listed. *type* specifies the port monitor type, for example, **listen**.

The command options function as follows:

`-l` By itself, the **-l** option lists status information for all services on the system.

`-l -p` *pmtag*
Lists status information for all services available through port monitor *pmtag*.

`-l -t` *type*
Lists status information for all services available through port monitors of type *type*.

Other combinations of options with **-l** are invalid.

The **-L** option is identical to the **-l** option except that its output is printed in a condensed format.

Options that request information write the requested information to the standard output. A request for information using the **-l** option prints column headers and aligns the information under the appropriate headings. A request for information in the condensed format using the **-L** option prints the information in colon-separated fields. If the **-l** option is used, empty fields are

indicated by a hyphen. If the **-L** option is used, empty fields are indicated by two successive colons.

The following sample output shows the differences between some of the options described above.

```
# sacadm  -l ↵
PMTAG    PMTYPE  FLGS RCNT STATUS   COMMAND
tcp      listen  -    3    ENABLED  /usr/lib/saf/listen tcp #listener for
tcp
ttymon1 ttymon   -    0    ENABLED  /usr/lib/saf/ttymon #
```

This is the most general form of the list option.

The STATUS field indicates the monitor's current state, whether disabled or enabled. Entries in the FLGS field do not change when a **ttymon** monitor changes from enabled to disabled or vice-versa. The FLGS field conveys information about the state in which a port monitor starts, not about its current state. For example, the **d** flag indicates that the port monitor goes immediately to DISABLED state when it is started.

The following command lists status information only for port monitor **tcp**:

```
# sacadm -l -p tcp ↵
PMTAG   PMTYPE  FLGS RCNT STATUS    COMMAND
tcp     listen  -    3    ENABLED   /usr/lib/saf/listen tcp #listener for
tcp
```

The same command using **-L** instead of **-l** will produce:

```
# sacadm -L -p tcp ↵
tcp:listen::3:DISABLED:/usr/lib/saf/listen -m tcp tcp # tcp listener
```

The following command lists status information for all port monitors whose type is **ttymon**:

```
# sacadm -l -t ttymon ↵
PMTAG     PMTYPE     FLGS RCNT STATUS     COMMAND
ttymon1   ttymon     -    0    ENABLED    /usr/lib/saf/ttymon #
```

Adding a port monitor

sacadm -a -p *pmtag* **-t** *type* **-c** *"cmd"* **-v** *ver* [**-f dx**] [**-n** *count*] \
[**-y** *"comment"*] [**-z** *script*]

The **sacadm** command with the **-a** option creates new instances of a port monitor. Because of the complexity of the options and

arguments that follow the **-a** option, it may be advisable for you to use a command script or the **sysadm** operation Device–> Port–> Port Monitor–> Add to add port monitors.

When **sacadm** creates a port monitor, it creates the supporting directory structure in **/etc/saf** and **/var/saf** for the new port monitor *pmtag* and the port monitor administrative file. It also adds an entry for the new port monitor to the SAC's administrative file.

The options following the **-a** option have the following meanings:

- The **-t** option is followed by the port monitor type. The type can be either **ttymon** or **listen**.
- The **-c** option is followed by a command enclosed in double quotes. This is the command SAC executes to start the port monitor.
- The **-v** option is followed by the version number of the port monitor. The version number may be given to **sacadm** by the port monitor's special administrative command, as an argument to the **-v** option. For example:

  ```
  -v `ttyadm -V`
  ```

  The port monitor-specific command is **ttyadm** for **ttymon** and **nlsadmin** for **listen** (see the manual pages **ttyadm**(1M) and **nlsadmin**(1M)). The version stamp of the port monitor is known by the command and is returned when the port monitor administrative command is invoked with the **-V** option. The version number is added to the new administrative file as a comment line of the form

  ```
  # VERSION=value
  ```

  where *value* is an integer that represents the port monitor version number. The version number defines the file format. It provides a means of synchronizing software releases of port monitors with their properly formatted administrative files.

- The **-f** option specifies one or both of the two flags **d** and **x**. The flags have the following meanings:

  d Do not enable the port monitor

  x Do not start the port monitor

  If the **-f** option is not included in the command line no flags are set and the default conditions prevail. By default a port monitor is started and enabled.

- The **-n** option sets the restart count to *count*. The restart determines how many times SAC will attempt to start the port monitor before placing it in the FAILED state. If a restart count is not specified when adding a port monitor, *count* is set to **0**. A count of **0** indicates that the port monitor is not to be restarted if it fails.

- The **-y** option includes "*comment*" in the SAC administrative file entry for the port monitor being added.
- The **-z** option names a file whose contents are installed as the per-port monitor configuration script, **_config**.

The command line below adds a port monitor of type **listen** (for TCP/IP).

```
# sacadm -a -p tcp -t listen -c "/usr/lib/saf/listen -m tcp tcp" \
   -v `nlsadmin -V` ↵
```

The following command line adds a port monitor of type **ttymon**.

```
# sacadm -a -p ttymon1 -t ttymon -c "/usr/lib/saf/ttymon" -v `ttymon -V` ↵
```

Enabling, disabling, starting, and stopping a port monitor

The commands to enable, disable, start, and stop a port monitor use the following syntax:

sacadm -e -p *pmtag*
sacadm -d -p *pmtag*
sacadm -s -p *pmtag*
sacadm -k -p *pmtag*

The **-e** option enables a port monitor. SAC sends an enable message to the port monitor.

The **-d** option disables a port monitor. SAC sends a disable message to the port monitor.

The **-s** option starts a port monitor.

The **-k** option kills a port monitor. SAC sends the signal SIGTERM to the port monitor.

Removing a port monitor

sacadm -r -p *pmtag*

Invoke **sacadm** with the **-r** option to remove port monitor *pmtag* from the system. The port monitor entry is removed from SAC's administrative file and SAC rereads the file. If the removed port monitor is not running, no further action is taken. If the removed port monitor is running, the Service Access Controller sends it a SIGTERM signal to indicate that it should shut down. Note that the

port monitor's directory structure remains intact but is no longer referenced by anything.

Printing, installing, and replacing configuration scripts

Per-system and per-port monitor configuration scripts are administered using **sacadm**; per-service configuration scripts are administered using **pmadm** and are described under "Service management." Per-system and per-port monitor configuration scripts allow you to modify the system and port monitor environments. They are written in the interpreted language described in the manual page for **doconfig**(3N). Sample configuration scripts are shown below.

The per-system configuration script, **_sysconfig**, is interpreted when SAC is starting. A port monitor's per-port monitor configuration script is interpreted by SAC just before SAC starts the port monitor.

Per-system and per-port monitor configuration scripts may be printed by any user on the system. Only someone with superuser privileges may install or replace them.

Per-system configuration scripts

sacadm -G [**-z** *script*]

The per-system configuration script **/etc/saf/_sysconfig** customizes the environment for all services on the system. When it starts up, the Service Access Controller interprets the per-system configuration script, using the **doconfig** library routine. The prototype **_sysconfig** file, **/etc/saf/_sysconfig.proto**, contains only a comment line.

The **-G** option is used to print or replace the per-system configuration script. The **-G** option by itself prints the per-system configuration script to the screen. The **-G** option in combination with the **-z** option replaces **/etc/saf/_sysconfig** with the contents of the file *script*. Other combinations of options with the **-G** option are invalid.

The sample **_sysconfig** file below sets the time zone variable, TZ.

```
assign TZ=EST5EDT          # set TZ
runwait echo SAC is starting > /dev/console
```

The **-z** option is also used with the **-a** option to specify the contents of the per-port monitor configuration file when a port monitor is created.

Per-port monitor configuration scripts

sacadm -g -p *pmtag* [**-z** *script*]

The per-port monitor configuration script **/etc/saf/***pmtag***/_config** customizes the environment for services that are available through the specific collection of access points for which port monitor *pmtag* is responsible. When SAC starts a port monitor, the per-port monitor configuration script is interpreted, if it exists, using the **doconfig**(3N) library routine.

The **-g** option is used to print, install, or replace a per-port monitor configuration script. A **-g** option requires a **-p** option. The **-g** option with only a **-p** option prints the per-port monitor configuration script for port monitor *pmtag*. The **-g** option with the **-p** option and a **-z** option installs the contents of file *script* as the per-port monitor configuration script for port monitor *pmtag*, or, if **/etc/saf/***pmtag***/_config** exists, it replaces **_config** with the contents of *script*. Other combinations of options with **-g** are invalid.

In the following sample **_config** file, the command **/usr/bin/daemon** is assumed to start a daemon process that builds and holds together a STREAMS multiplexor. By installing this configuration script, the command can be executed just before starting the port monitor that requires it.

```
run /usr/bin/daemon
# build a STREAMS multiplexor
runwait echo $PMTAG is starting > /dev/console
```

Reading the administrative files

sacadm -x [**-p** *pmtag*]

When changes are made to SAC's administrative file, SAC needs to be notified of the change. When changes are made to a port monitor's administrative files, the port monitor needs to be notified. When **sacadm** and **pmadm** are used to make changes, this notification takes place automatically. Therefore we recommend that you change administrative files with the commands and avoid changing the file by editing it directly.

If you do edit the administrative files, SAC and the port monitors are not notified. In this case, you must notify SAC and the port monitors about the changes by running **sacadm** with appropriate options. The **-x** option tells SAC to update its internal copy of the information in the SAC administrative file. **sacadm** with the **-x** and **-p** options causes SAC to send a READ message to the designated port monitor.

Service management

The top level of SAF is concerned with port monitor administration and is discussed in "Port monitor management." The lower level is concerned with service administration and is discussed in this section.

At this level there are two distinct administrative functions. The first is the administration of the port itself. The information needed to administer a terminal port will be found on the manual page for **ttymon**'s port monitor-specific command, **ttyadm**(1M). The information needed to administer a network address monitored by a **listen** port monitor will be found on the manual page for **listen**'s port monitor-specific command, **nlsadmin**(1M).

The second is the administration of the service associated with a port. By definition, there is one and only one service associated with a port. All ports on the system are peers and their services are administered through the same command interface, the Service Access Facility's administrative command **pmadm**(1M). At the level of service administration, services may be added, removed, enabled, and disabled. Other functions performed at this level include installing or replacing a per-service configuration script and requesting service status information.

The port monitor administrative command: pmadm

pmadm is the administrative command for the lower level of the Service Access Facility hierarchy, that is, for service administration. A port may have only one service associated with it although the same service may be available through more than one port. **pmadm** performs the following functions:

- Print service status information from the port monitor's administrative file
- Add or remove a service
- Enable or disable a service
- Install or replace a per-service configuration script

Note that in order to identify an instance of a service uniquely, the **pmadm** command must identify both the service (**-s**) and the port monitor or port monitors through which the service is available (**-p** or **-t**).

Printing service status information

```
pmadm -l  [ -t type | -p pmtag ] [ -s svctag ]
pmadm -L  [ -t type | -p pmtag ] [ -s svctag ]
```

The **–l** and **–L** options request service status information. They may be invoked by any user on the system. Used either alone or with the options described below they provide a filter for extracting information in several different groupings.

–l
By itself, the **–l** option lists status information for all services on the system.

–l –p *pmtag*
Lists status information for all services available through port monitor *pmtag*.

–l –s *svctag*
Lists status information for all services with the tag *svctag* available through any port monitor on the system.

–l –p *pmtag* **–s** *svctag*
Lists status information for service *svctag* available through port monitor *pmtag*.

–l –t *type*
Lists status information for all services available through port monitors of type *type*.

–l –t *type* **–s** *svctag*
Lists status information for all services with the tag *svctag* offered through a port monitor of type *type*.

Other combinations of options with **–l** are invalid.

The **–L** option is identical to the **–l** option except that output is printed in a condensed format.

Options that request information write the requested information to the standard output. A request for information using the **–l** option prints column headers and aligns the information under the appropriate headings. A request for information in the condensed format using the **–L** option prints the information in colon-separated fields. If the **–l** option is used, empty fields are indicated by a hyphen. If the **–L** option is used, empty fields are indicated by two successive colons.

The example below shows a sample of **–l** output. The lines have been broken for readability.

```
PMTAG PMTYPE SVCTAG FLGS ID    PMSPECIFIC
tmon3 ttymon 6      ux   root  /dev/tty06 - - /usr/bin/login - 9600 - login: - \
                                #tty6
tmon3 ttymon 7      ux   root  /dev/tty07 - - /usr/bin/login - 9600 - login: - \
                                #tty7
tmon3 ttymon 8      ux   root  /dev/tty08 - - /usr/bin/login - 9600 - login: - \
                                #tty8
tmon3 ttymon 9      ux   root  /dev/tty09 - - /usr/bin/login - 9600 - login: - \
                                #tty9
tmon1 ttymon 1      ux   root  /dev/tty01 - - /usr/bin/login - 9600 - login: - \
                                #tty1
tmon1 ttymon 2      ux   root  /dev/tty02 - - /usr/bin/login - 9600 - login: - \
                                #tty2
tmon1 ttymon 3      ux   root  /dev/tty03 - - /usr/bin/login - 9600 - login: - \
                                #tty3
tmon1 ttymon 4      ux   root  /dev/tty04 - - /usr/bin/login - 9600 - login: - \
                                #tty4
tcp   listen 101    -    listen - c - /usr/lib/uucp/uucico -r0 -unuucp -iTLI \
                                #UUCP access direct to server
tcp   listen 1000   -    listen - c - /usr/tcp/lib/tstserver #TP TEST SERVER
tcp   listen 0      -    root   - c - sfbul.serve c - /usr/lib/saf/nlps_server
```

Adding a service

pmadm -a [**-p** *pmtag* | **-t** *type*] **-s** *svctag* **-i** *id* **-m** *"pmspecific"* **-v** *ver* ****
[**-f xu**] [**-y** *"comment"*] [**-z** *script*]

The **-a** option adds a service by making an entry for the new service in the port monitor's administrative file. It is important to be aware that a service implies a port and that there is a one-to-one mapping between ports and instances of services. Because of the complexity of the options and arguments that follow the **-a** option, it may be advisable to use a command script or **sysadm** to add services. If you use **sysadm**, select the operation Device–> Port–> Port Service–> Add.

The following paragraphs describe the components of the command line for adding a service.

-p *pmtag*
Specify the tag for the port monitor. This option causes **pmadm** to add the service to the port monitor designated as *pmtag*. This is a name you select.

-s *svctag*
Specify the tag for the port service. This is a name you select.

-t *pmtype*
Specify a group of port monitors by port monitor type. You

may not specify both this option and the **-p** option. This option adds the service to all port monitors of type *type*.

-i *login*
Specify the user login name that will own the port service process. The login name must already exist.

-m *options*
Use this option to specify port monitor-specific options for **pmadm**. To generate the options, embed a port monitor-specific command, delimited with shell backquote characters (`), as an argument to the **-m** option. The port monitor-specific command for **ttymon** is **ttyadm**(1M). The port monitor-specific command for **listen** is **nlsadmin**(1M).

-v *version*
The version of the port monitor administrative file. For a port monitor of type **listen**, for example, the version number may be given as

```
-v `nlsadmin -V`
```

The version stamp of the port monitor is known by the port monitor-specific command and is returned when the command is invoked with the **-V** option.

-f Specify one or both of two flags which are then included in the flags field of the port monitor administrative file entry for the new service. The flags have these meanings:

x Do not enable the service.

u Create a **utmp** entry for the service.

If the **-f** option is not included on the **-a** command line, no flags are set and the default conditions prevail. By default, a new service is enabled and no **utmp** entry is created for it.

-y *"comment"*
Specify a comment in double quotes. The comment is included in the comment field for the service entry in the port monitor administrative file.

-z *script*
Installs *script* as a configuration file.

The following example adds a service with service tag **105** to all port monitors of type **listen**. The line is broken for readability.

```
# pmadm -a -s 105 -i root -t listen -v `nlsadmin -V` \
          -m `nlsadmin -a 105 -c /usr/net/servers/rfsetup` ↵
```

Enabling or disabling a service

```
pmadm -e -p pmtag -s svctag
pmadm -d -p pmtag -s svctag
```

The **-e** option enables a service. **x** is removed from the flags field in the entry for service *svctag* in the port monitor administrative file.

The **-d** option disables a service. **x** is added to the flags field in the entry for service *svctag* in the port monitor administrative file.

Removing a service

```
pmadm -r -p pmtag -s svctag
```

The **-r** removes service *svctag*. The entry for the service is removed from the port monitor administrative file.

Printing, installing, and replacing per-service configuration scripts

```
pmadm -g -p pmtag -s svctag [ -z script ]
pmadm -g -s svctag -t type -z script
```

Per-service configuration scripts are command scripts written in the interpreted language described in the **doconfig**(3N) manual page. They allow you to modify the environment in which a service executes. For example, the values of environment variables may be changed, STREAMS modules may be specified, or commands may be run.

Per-service configuration scripts are interpreted by the port monitor before the service is invoked. SAC interprets both its own configuration file, **_sysconfig**, and the port monitor configuration scripts. Only the per-service configuration scripts are interpreted by the port monitors. Per-service configuration scripts may be printed by any user on the system. Only someone with superuser privileges can install or replace them.

The **-g** option is used to print, install, or replace a per-service configuration script. The **-g** option with a **-p** option and a **-s** option prints the per-service configuration script for service *svctag* available through port monitor *pmtag*. The **-g** option with a **-p** option, a **-s** option, and a **-z** option installs the per-service configuration script contained in the file *script* as the per-service configuration script for service *svctag* available through port monitor *pmtag*. The **-g** option with the **-s** option, a **-t** option, and a

–z option installs the file *script* as the per–service configuration script for service *svctag* available through any port monitor of type *type*. Other combinations of options with **–g** are invalid.

Examine the following sample per-service configuration script:

```
runwait ulimit 4096
runwait umask 077
```

This script does two things. It specifies the maximum file size for files created by a process by setting the process's **ulimit** to **4096**. It also specifies the protection mask to be applied to files created by the process by setting **umask** to **077.**

The port monitor: ttymon

ttymon is a port monitor invoked by SAC, the controlling process for SAF. It is started by **init** when the system enters multiuser mode. One of SAC's functions is to start all of the port monitors that you configured.

ttymon sets terminal modes and line speeds for the port the user is connected to, allowing communication with the service associated with that port.

What ttymon does

ttymon has three main functions:

- Initializes and monitors TTY ports
- Sets terminal modes and line speeds for each port it monitors
- Invokes the service associated with a given port whenever it receives a connection request on that port

Each instance of **ttymon** has its own administrative file that specifies the ports to monitor and the services associated with each port. The file contains a *ttylabel* field that refers to a speed and TTY definition in the **/etc/ttydefs** file. See **ttyadm** (1M) for a description of the information specific to **ttymon** that is contained in a **ttymon** administrative file.

When a **ttymon** port monitor is started, it initializes all ports specified in its administrative file, pushes the specified STREAMS modules onto the ports, sets speed and initial **termio**(7) settings, and writes the prompt to the port. It then waits for user input.

A connection request is successful when at least one non-break character followed by a New Line character is received from the

port. If the service to be invoked is **login**, the New Line character will be preceded by the users login name. A New Line character will not be recognized unless the line speed of the port and the line speed of the device connected to the port are the same.

If an unreadable prompt is printed on the terminal, the user sends a BREAK to indicate that the port and device line speeds are not compatible. See your terminal documentation to see how to generate a BREAK signal. Each break indication will cause **ttymon** to hunt to the next *ttylabel* in **/etc/ttydefs**, adjusting its **termio**(7) values and reissuing the prompt.

On successful completion of the connection request, **ttymon** interprets the per-service configuration script, if one exists. It then invokes the service associated with the port. A typical example of a service associated with a port is **login**.

ttymon has no interaction with its TTY ports while they are connected to a service. On completion of a service on a port, **ttymon** returns the port to its initial state.

The autobaud option

Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention. Each time a service to be monitored by a **ttymon** port monitor is added, a *ttylabel* must be supplied (see "Adding a service"). If this *ttylabel* points to an entry in the **/etc/ttydefs** file that has an **A** in the autobaud field, **ttymon** will try to determine the proper line speed before printing the prompt.

After receiving a carrier-indication on one of its TTY ports, but before printing a prompt, **ttymon** does the following:

- **ttymon** reads the next character received from the port. Provided the character read is a New Line character and that it is transmitted at a line speed autobaud can support, **ttymon** will reliably determine this line speed and change the port's line speed to that speed.
- If a baud rate cannot be determined from the character that is read (for example, if the user entered a character other than a New Line), or if a break is received rather than a character, **ttymon** considers this to be an autobaud failure and the character is discarded. If after five opportunities, a New Line is not recognized, the search proceeds to the next **ttydefs** entry in the hunt sequence. If an autobaud flag is encountered again, the prompt will not be written and the procedure just described is repeated. If no autobaud flag is set, the search again proceeds to the next **ttydefs** entry in the hunt sequence.

ttymon and SAF

SAF provides a generic interface to which all port monitors must conform. **ttymon** is a port monitor under the SAF's controller, SAC.

There can be multiple invocations of **ttymon** port monitors, each identified by a unique *pmtag*. Each of these port monitors can monitor multiple ports for incoming connection requests.

A port has one and only one service associated with it. Each port and its associated service are identified by a service tag, *svctag*. Service tags for any given port monitor are unique.

When SAC starts a port monitor, the port monitor reads its administrative file, which contains information about which ports to monitor and what service (that is, process) is associated with each port.

The default ttymon configuration

Some **ttymon** port monitors may be set up automatically when the system goes to multiuser level. To find out if your system has been automatically configured, enter the command:

```
# sacadm -l ↵
```

after the system is in multiuser mode. To see a listing of all services available under the configured **ttymon** port monitors, enter the command:

```
# pmadm -l -t ttymon ↵
```

The line discipline module, **ldterm**, may not be specified for automatically configured services. Instead, it may be defined in an autopush administrative file and pushed by the autopush facility (see the **autopush**(1M) manual page). **autopush** pushes previously specified modules onto the appropriate STREAM each time a device is opened.

Services are not defined for the console and contty ports under any **ttymon** port monitor. Instead, there is an entry for each service in the **/etc/inittab** file. These entries contain calls to **ttymon** in "express." See "ttymon express mode" for information.

The ttyadm command

SAF requires each type of port monitor to provide an administrative command. This command must format information derived from command-line options so that it is suitable for inclusion in the

administrative files for that port monitor type. The command may also perform other port monitor-specific functions.

ttyadm is **ttymon**'s administrative command. The **ttyadm** command formats information based on the options with which it is invoked and writes this information to the standard output.

ttyadm is one of the arguments **pmadm** uses with the **-a** option to format information in a way suitable for inclusion in a **ttymon** administrative file. **ttyadm** presents this information (as standard output) to **pmadm**, which places it in the file. This use of **ttyadm** is described under "Adding a ttymon port monitor." Port monitor-specific information in a port monitor administrative file will be different for different port monitor types.

ttyadm is also included on the **sacadm** command line when a port monitor is added to the system. It is used to supply the **ttymon** version number for inclusion in a port monitor's administrative file. The port monitor administrative file is updated by the Service Access Controller's administrative commands, **sacadm** and **pmadm**. **ttyadm** merely provides a means of presenting formatted port monitor-specific (that is, **ttymon**-specific) data to these commands. The **sacadm** command line uses **ttyadm** only with the **-V** option. **ttyadm -V** tells SAC the version number of the **ttymon** command being used.

Managing TTY ports

This section discusses and shows examples of port, port monitor, and service management. It includes the syntax of relevant commands.

Listing configured ttymon port monitors

sacadm -l [**-p** *pmtag* | **-t** *type*]

The **sacadm** command with only a **-l** option lists all port monitors currently defined for the system. The following is an example of its output:

```
PMTAG     PMTYPE    FLGS RCNT STATUS    COMMAND
tcp       listen    -    3    ENABLED   /usr/lib/saf/listen tcp #listener for tcp
ttymon1   ttymon    -    0    ENABLED   /usr/lib/saf/ttymon #
```

sacadm can also be used to list a single port monitor (**-p**) or to list only port monitors of a single type (**-t**), for example, all port monitors of type **ttymon**. For a complete description of these options, see "Printing port monitor status information" in "Port monitor management," or see the **sacadm**(1M) manual page.

Listing services configured for a ttymon port monitor

pmadm -l [**-p** *pmtag* | **-t** *type*] [**-s** *svctag*]

pmadm with only a **-l** will list all services for all port monitors on the system. If a port monitor is specified (**-p**), all services for that port monitor will be listed. The following is an example listing for the command:

```
# pmadm -l -p ttymon2 ↵

PMTAG    PMTYPE SVCTAG FLGS ID   PMSPECIFIC
ttymon2 ttymon 21     ux   root /dev/tty21 -- /usr/bin/login - 9600 - login: -
ttymon2 ttymon 22     ux   root /dev/tty22 -- /usr/bin/login - 9600 - login: -
ttymon2 ttymon 23     ux   root /dev/tty23 -- /usr/bin/login - 9600 - login: -
ttymon2 ttymon 24     ux   root /dev/tty24 -- /usr/bin/login - 9600 - login: -
```

In the example, the *pmspecific* fields include the device (for example, **/dev/tty21**), the service to be invoked (**/usr/bin/login**), and the prompt (`login:`). See the **ttyadm**(1M) manual page for a description of the *pmspecific* fields.

Listing accessible TTY ports

To find out which ports are accessible to users, first identify all enabled **ttymon** port monitors:

```
# sacadm -l -t ttymon ↵
PMTAG     PMTYPE     FLGS RCNT STATUS     COMMAND
ttymon1   ttymon     -    0    ENABLED    /usr/lib/saf/ttymon #
```

In the listing, port monitor **ttymon1** is enabled. This means that it is accepting service requests for any of its services that are enabled.

To identify which services are enabled, use **pmadm -l -p ttymon1**. This will list all configured TTY services for port monitor **ttymon1** as in the following example:

```
# pmadm -l -p ttymon1 ↵
PMTAG    PMTYPE SVCTAG FLGS ID  <PMSPECIFIC>
ttymon1 ttymon 11     u    root /dev/tty11 - - /usr/bin/login - 9600 - login: -
ttymon1 ttymon 12     ux   root /dev/tty12 - - /usr/bin/login - 9600 - login: -
ttymon1 ttymon 13     u    root /dev/tty13 - - /usr/bin/login - 9600 - login: -
ttymon1 ttymon 14     ux   root /dev/tty14 - - /usr/bin/login - 9600 - login: -
```

In the listing, enabled services are those that do *not* have an **x** in the FLGS column. The ports corresponding to these services (**/dev/tty11** and **/dev/tty13**) are accessible to users. The **who -l**

command lists all running port monitors, not the accessible TTY ports. Follow the procedure described above to find out which TTY ports are accessible.

Adding a ttymon port monitor

```
sacadm -a -p pmtag -t type -c cmd -v `ttyadm -V` \
                -n count [ -f dx ] [ -z script ] [ -y comment ]
```

The following command line will add a **ttymon** -type port monitor named **ttymon1**:

```
# sacadm -a -p ttymon1 -t ttymon -c /usr/lib/saf/ttymon \
       -v `ttyadm -V` ↵
```

The command adds a line to SAC's administrative file. The options that may be used with **sacadm -a** are described under "Port monitor management" and in the **sacadm**(1M) and **ttyadm**(1M) manual pages. If a port monitor already exists with the same name as the port monitor that is being added, you must remove the old port monitor before adding the new one.

Removing a ttymon port monitor

```
sacadm -r -p pmtag
```

The following command line removes the port monitor added in the previous example:

```
# sacadm -r -p ttymon1 ↵
```

SAC removes the line for port monitor **ttymon1** from its administrative file. The port monitor directory will remain in **/etc/saf** but will be removed and recreated when a new port monitor with the same name is added. To make changes to a port monitor entry, always remove the entry and add a new entry using **sacadm** or **sysadm**.

IMPORTANT: Do not edit the SAC administrative file. If you edit the file, you could introduce format or syntax errors that could affect the function of your port monitors in undesirable ways.

Adding a service

```
pmadm -a -p pmtag -s svctag -i id [ -f ux ] -v `ttyadm -V` -m "`ttyadm \
    [ -b ] [ -r count ] [ -c ] [ -h ] [ -i msg ] [ -m modules ] \
    [ -p prompt ] [ -t time-out ] -d device -l ttylabel -s service `"
```

The following command line adds a login service to be monitored by the **ttymon** port monitor **ttymon2**:

```
# pmadm -a -p ttymon2 -s 21 -i root -fu -v `ttyadm -V` -m"`ttyadm -d \
    /dev/tty21 -l 9600 -s /usr/bin/login -m ldterm -p \"tty21:\"`" ↵
```

The options that may be used with **pmadm -a** are described under "Service management" and on the **pmadm**(1M) and **ttyadm**(1M) manual pages.

The **ttyadm -m** option may be used for pushing STREAMS modules, for example the line discipline module, **ldterm**. If **autopush** has pushed modules on the stream, **ttymon** pops them before pushing its own.

By using the **ttyadm -i** option, we could also have specified a message to be printed whenever someone tries to log in on a disabled port.

The following command defines a service that permits both incoming and outgoing calls. The service is put under port monitor **ttymon2**. The **-b** option defines the port as bi-directional.

```
# pmadm -a -p ttymon2 -s 21 -i root -fu -v `ttyadm -V` \
     -m "`ttyadm -b -h -r0 -t 60 -d /dev/tty21 \
     -l 9600H -s /usr/bin/login -m ldterm -p \" tty21:\"`" ↵
```

The **ttyadm -r** option with **count=0** is assumed when the **ttyadm -b** bi-directional option is used; the **-r0** could therefore have been omitted.

Removing a service

pmadm -r -p *pmtag* **-s** *svctag*

The following example deletes the service that was added in the previous example.

```
# pmadm -r -p ttymon2 -s 21 ↵
```

Enabling a service

pmadm -e -p *pmtag* **-s** *svctag*

To enable a service on a specific port, first find out which port monitor is monitoring the port. Enter:

```
# pmadm -l -t ttymon ↵
```

This lists all services defined for ttymon-type ports.

Now look in the PMSPECIFIC column for the device file that corresponds to the port you are interested in, for example, **/dev/tty23**. If the port monitor is **ttymon2** and the service tag is **23**, the command:

```
# pmadm -e -p ttymon2 -s 23 ↵
```

will enable the service on port **/dev/tty23**.

To verify that the port has been enabled, enter:

```
# pmadm -l -p ttymon2 -s 23 ↵
```

The **x** will have been removed from the FLGS column in the entry for this service.

Disabling a service

pmadm -d -p *pmtag* **-s** *svctag*

When a service is disabled, all subsequent connection requests for the service will be denied. Using the same example:

```
# pmadm -d -p ttymon2 -s 23 ↵
```

will restore the **x** to the FLGS field in the entry for service **23**.

Disabling all Services monitored by a ttymon port monitor

sacadm -d -p *pmtag*

To disable all services defined for the port monitor **ttymon2**, enter:

```
# sacadm -d -p ttymon2 ↵
```

Any future connection requests for services managed by this port monitor will be denied until the port monitor is enabled.

The command:

```
# sacadm -e -p ttymon2 ↵
```

re-enables port monitor **ttymon2**.

ttymon express mode

Services are not defined for the console and TTY ports under any **ttymon** port monitor. Instead, there is an entry for each service in the **/etc/inittab** file. These entries contain calls to **ttymon** in "express" mode. **ttymon** express is a special mode of **ttymon** that permits **ttymon** to be invoked directly by a command that requires login service. **ttymon** in express mode is not managed by SAC nor is an administrative file associated with any invocation of **ttymon** in this mode.

ttymon express is described in greater detail on the **ttymon**(1M) manual page.

Per-service configuration scripts

As a port monitor under SAF, **ttymon** can customize the environment of each service it starts. It does this by interpreting a per-service configuration script, if one exists, immediately before starting the service. Per-service configuration scripts are optional. You install configuration scripts using the **pmadm** command with **-g** and **-z** options (see the **pmadm**(1M) manual page).

It is also possible to customize the environment of a **ttymon** port monitor. A per-port monitor configuration script is defined using the **sacadm** command with **-g** and **-z** options (see the **sacadm**(1M) manual page). The environment modifications made by a port-monitor configuration script are inherited by the port monitor and all the services it invokes. The environment of any particular service can then be customized further by using a per-service configuration script.

The **doconfig**(3N) manual page describes the language in which configuration scripts are written.

Configuration scripts are not normally needed for basic operations.

The who command

The **who** command examines the **/etc/utmp** file. It is used to find out who is on the system. The following command lists all entries in the **utmp** file including all **RUNNING** port monitors:

```
# who -lH  ↵

NAME     LINE     TIME          IDLE  PID   COMMENTS
LOGIN    contty   Jun 17 12:49  old   8226
tcp      .        Jun 17 12:50  old   8230
ttymon1  .        Jun 17 12:50  old   8234
ttymon3  .        Jun 17 12:50  old   8235
```

When **ttymon** in express mode is monitoring a line, the name field is **LOGIN** as it is in the entry for **contty** in the preceding example.

The following command lists all users who are currently logged in:

```
# who -u ↵
root       console      Jun 17 13:07    .       8303
john       term/32      Jun 17 13:13   0:01     8353
```

If **ttymon** invokes a service other than **login**, an entry for this service will appear beginning with a "**\.**" and giving the terminal line. To find out which ports are accessible but currently not in use, use the following command:

```
# pmadm -l -t ttymon ↵
```

Identifying ttymon processes

The **ps** command lists all processes. Since **ttymon** port monitors fork a process to handle each connection request, the number of **ttymon**-related entries that appear in the output of a **ps** listing may be greater than the number of running **ttymon** port monitors.

When a **ttymon** port monitor forks a child to process a connection request (that is, to do baud rate searching, set final **termio** options, and so on, before invoking the service), the port will be identified in the TTY field for this child process. For the parent **ttymon** port monitor process, this TTY field will be empty.

Log files

Problems often arise when a single port is monitored by more than one process. If a port (for example, **/dev/tty11**) is used by an enabled service under a **ttymon** port monitor running under the Service Access Facility, and the same port is also monitored by a **ttymon** process running in **ttymon** express mode, (that is, started by **init** when it reads **inittab**, not by **sac** when it reads its administrative file) then the port will behave unpredictably. You are expected to examine the system for such ambiguously configured ports.

There are also two log files that can be examined for clues to problems related to **ttymon** port monitors or ports monitored by **ttymon** port monitors. SAC records aberrant port monitor behavior in the **/var/saf/_log** file. Each **ttymon** port monitor also has its own log file, **/var/saf/***pmtag***/log**, where it records messages it receives from SAC, services it starts, and so on.

The following command:

```
# tail  -25  /var/saf/_log ↵
```

lists the most recent 25 entries in the **_log** file.

Periodically, you should truncate the log files. You can set up a **cron** job to clean up regularly. See Chapter 6 for how to automate jobs with **cron** and to perform log cleanup. Also see the manual pages for **cron**(1M) and **crontab**(1).

Terminal line settings

init is a general process spawner that is invoked as the last step in the boot procedure. It starts SAC. SAC then looks in its administrative file to see which port monitors to start. Each **ttymon** port monitor started by SAC looks in *its* administrative file for the TTY ports to initialize.

For each TTY port initialized, **ttymon** searches the **ttydefs** file for the information it needs to set terminal modes and line speeds. **ttymon** then waits for service requests. When a service request is received, **ttymon** executes the command (usually **login**) associated with the port that received the request. This command is contained in the entry for the port in the port monitor's administrative file.

The key elements in managing terminal line settings are the **ttydefs** file and the **sttydefs** command, which maintains the **ttydefs** file.

The sttydefs command

sttydefs(1M) is an administrative command that maintains the **ttydefs** file. The **ttydefs** file contains information about line settings and hunt sequences for the system's TTY ports. The **sttydefs** command and the **ttydefs** file together provide the facilities for managing terminal modes and line settings. The **sttydefs** command is used to:

- Print information contained in **ttydefs**
- Add records for terminal ports to the **ttydefs** file
- Remove records from the **ttydefs** file

Printing terminal line setting Information

```
/usr/sbin/sttydefs  -l  [ttylabel]
```

If a *ttylabel* is specified, **sttydefs** prints the **ttydefs** record that corresponds to this *ttylabel*. If no *ttylabel* is specified, **sttydefs** prints this information for all records in the **/etc/ttydefs** file. **sttydefs** verifies that each entry it displays is correct and that the entry's *nextlabel* field refers to an existing *ttylabel*. An error message is printed for each invalid entry detected.

Adding records to the ttydefs file

```
/usr/sbin/sttydefs -a ttylabel [-b] [-n nextlabel] [-i initial-flags] \
          [-f final-flags]
```

sttydefs with the **-a** option adds a record to the **ttydefs** file.

ttylabel identifies the record.

The following list describes the effect of the **-b**, **-n**, **-i**, or **-f** options when used with the **-a** option. The **-a** option is valid only when invoked by a privileged user.

-b Enables autobaud.

-n Specifies the value to be used in the *nextlabel* field. If *nextlabel* is not specified, **sttydefs** will set *nextlabel* to *ttylabel*.

-i Specifies the value to be used in the *initial-flags* field. The argument to this option must be presented in a format recognized by the **stty** command. If *initial-flags* is not specified, **sttydefs** will set *initial-flags* to the **termio**(7) flag **9600**.

-f Specifies the value to be used in the *final-flags* field. The argument to this option must be presented in a format recognized by the **stty** command. If *final-flags* is not specified, **sttydefs** will set *final-flags* to the **termio**(7) flags **9600** and **sane**.

The following command line creates a new record in **ttydefs**:

```
# sttydefs -aNEW -nNEXT -i"1200 hupcl erase ^h" -f"1200 sane ixany \
          hupcl erase ^h echoe" ↵
```

The flag fields shown have the following meanings:

300–19200
The baud rate of the line.

hupcl Hang up on close.

sane A composite flag that stands for a set of normal 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.

erase ^h
Set the erase character to **^h**. On most terminals a **^h** is the backspace.

echoe Echo erase character as the string backspace-space-backspace. On most terminals this will erase the erased character.

Creating a hunt sequence

The following sequence of commands adds records with labels **1200**, **2400**, **4800**, and **9600** to the **ttydefs** file and puts them in a circular list or hunt sequence:

```
# sttydefs -a1200 -n2400 -i 1200 -f "1200 sane" ↵
# sttydefs -a2400 -n4800 -i 2400 -f "2400 sane" ↵
# sttydefs -a4800 -n9600 -i 4800 -f "4800 sane" ↵
# sttydefs -a9600 -n1200 -i 9600 -f "9600 sane" ↵
```

The *nextlabel* field of each line is the *ttylabel* of the next line. The *nextlabel* field for the last line shown points back to the first line in the sequence.

The object of a hunt sequence is to link a range of line speeds. Entering a BREAK during the baud rate search causes **ttymon** to step to the next entry in the sequence. See your terminal documentation to see how to generate a BREAK signal. The hunt continues until the baud rate of the line matches the speed of the user's terminal.

The **ttydefs** file containing these records will look like this:

```
# VERSION=1
1200:1200:1200 sane::2400
2400:2400:2400 sane::4800
4800:4800:4800 sane::9600
9600:9600:9600 sane::1200
```

Removing records from the ttydefs file

/usr/sbin/sttydefs -r *ttylabel*

The record for the *ttylabel* specified on the command line is removed from the **ttydefs** file.

The **–r** option is valid only when invoked by a privileged user. If a record you remove is part of a hunt sequence, be sure the sequence is repaired. It may be useful to run **sttydefs** with the **–l** option after a record has been removed. **sttydefs –l** will check for incorrect field values and broken hunt sequences and will print error messages.

The ttydefs file

/etc/ttydefs is an administrative file used by **ttymon**. It defines speed and terminal settings for TTY ports. The **ttydefs** file contains the information listed below. Figure 11–2 shows the relationship between the *ttylabel* and *nextlabel* fields in the **ttymon** administrative files and **ttydefs** files. The example after that shows a sample **ttydefs** file.

ttylabel

When **ttymon** initializes a port, it searches the **ttydefs** file for the entry that contains the **termio**(7) settings for that port. The correct entry is the one whose *ttylabel* matches the *ttylabel* for the port. The *ttylabel* for the port is part of the *pmspecific* information included in **ttymon**'s administrative file. By convention, **ttylabel** identifies a baud rate (for example, **1200**), but it need not.

initial–flags

Contains the **termio**(7) options to which the terminal is initially set. *initial–flags* must be specified using the syntax recognized by the **stty**(1) command.

final–flags

Contains the **termio**(7) options set by **ttymon** after a connection request has been made and immediately before invoking a port's service. *final–flags* must be specified using the syntax recognized by **stty**.

autobaud

autobaud is a line-speed option. When *autobaud* is used instead of a baud rate setting, **ttymon** determines the line speed of the TTY port by analyzing the first carriage return entered and sets the speed accordingly. If the *autobaud* field contains the character **A**, the *autobaud* facility is enabled; otherwise, *autobaud* is disabled.

nextlabel

If the user indicates (by sending a BREAK) that the current **ttydefs** entry does not provide a compatible line speed, **ttymon** will search for the **ttydefs** entry whose *ttylabel* matches the *nextlabel* field. **ttymon** will then use that field as its *ttylabel* field. A series of speeds is often linked together in this way into a closed set called a hunt

sequence. For example, **4800** may be linked to **1200**, which in turn is linked to **2400**, which is finally linked to **4800**.

All **termio**(7) settings supported by the **stty** command are supported as options in the **ttydefs** file. For example, you will be able to specify the default erase and kill characters.

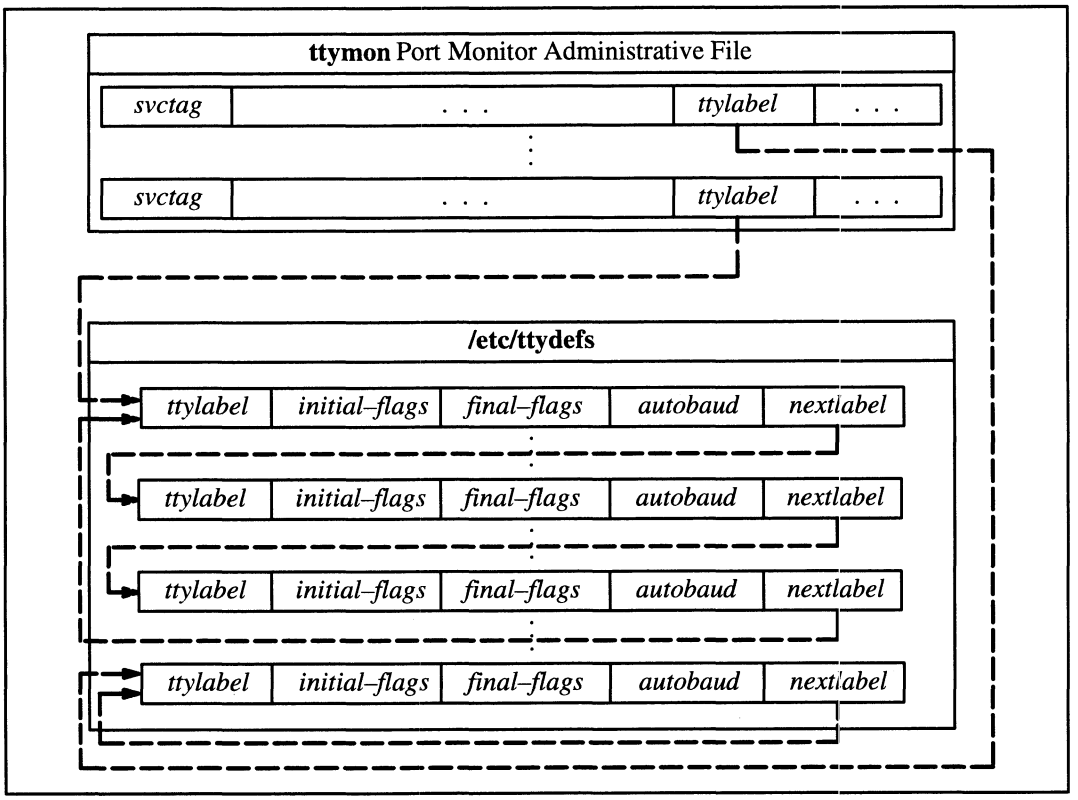


Figure 11–2 Port monitor/ttydefs links

The format of the **/etc/ttydefs** file may change in future releases. For continuity across releases, use the **sttydefs**(1M) command to access this file. Following is a sample **ttydefs** file:

```
# VERSION=1
38400:38400 hupcl erase ^h:38400 sane ixany tab3 hupcl erase ^h::19200
19200:19200 hupcl erase ^h:19200 sane ixany tab3 hupcl erase ^h::9600
9600:9600 hupcl erase ^h:9600 sane ixany tab3 hupcl erase ^h::4800
4800:4800 hupcl erase ^h:4800 sane ixany tab3 hupcl erase ^h::2400
2400:2400 hupcl erase ^h:2400 sane ixany tab3 hupcl erase ^h::1200
1200:1200 hupcl erase ^h:1200 sane ixany tab3 hupcl erase ^h::300
300:300 hupcl erase ^h:300 sane ixany tab3 hupcl erase ^h::19200
```

Setting terminal options with the stty command

The **stty**(1) command may be used to set or change terminal options after a user has logged in. A **stty** command line may also be added

to a user's **.profile** to set options automatically as part of the **login** process. The following is an example of a simple **stty** command:

```
# stty cr0 nl0 echoe -tabs erase ^h ↵
```

The options in the example have the following meanings:

cr0 nl0
: No delay for carriage return or new line. Delays are not used on a video display terminal, but are necessary on some printing terminals to allow time for the mechanical parts of the equipment to move.

echoe Erase characters as you backspace.

–tabs Expand tabs to spaces when printing.

erase ^h
: Change the character-delete character to **^h**. The default character-delete character is the number sign (#). Most terminals transmit a **^h** when the **backspace** key is pressed.

The port monitor: listen

listen is a port monitor invoked by SAC, the controlling process for SAF. It is started by **init** when the system enters multiuser mode. One of SAC's functions is to start all of the port monitors you configured.

listen monitors a connection-oriented transport network, receiving incoming connection requests, accepting them, and invoking the services that have been requested. The **listen** port monitor may be used with any connection-oriented transport provider that conforms to the Transport Interface (TLI) specification. The TLI is documented in *Programming with TCP/IP on the DG/UX™ System*.

What listen does

The **listen** port monitor performs functions common to all port monitors:

- Initializes and monitors **listen** ports
- Invokes the service associated with a port in response to requests

The **listen** port monitor also has these features:

- Allows private addresses for services
- Passes connections (file descriptors) to standing servers
- Supports socket-based services
- Supports RPC-based services and dynamic addressing

Private addresses for services

Each **listen** service may have a transport address in addition to its service code (*svctag*). This private address is included in the port monitor's administrative file. The inclusion of private addresses for services allows a single **listen** process to monitor multiple addresses. The number of addresses that **listen** can listen on is determined by the number of file descriptors available to the process.

Passing connections to standing servers

By default, a new instance of a service is invoked for each connection. This feature is useful for server processes that need to maintain state information. A standing server is a server process or service that runs continuously and accepts connections through a FIFO or a named STREAM instead of being propagating copies of itself with calls to **fork**(2) and **exec**(2).

Socket-based services

listen supports services that use sockets as their interface to the transport provider. Socket-based services are registered with **listen** in the same way TLI services are, using the Service Access Facility's administrative commands. **listen** supports STREAMS; sockets are implemented as a STREAMS module and a library.

A socket-based service:

- May or may not be an RPC service
- May have a statically or dynamically assigned address, or no private address
- May be invoked on each connection or may be a standing server, to which new connections are passed by a FIFO or a named STREAM

RPC services and dynamic addressing

Dynamic addressing is most useful with RPC. RPC transport addresses may be either specified or dynamically assigned. In either case, **listen** tells the **rpcbinder** what the address is and monitors it for incoming connections.

In the case of a dynamically assigned address, **listen** asks the transport provider to select a transport address each time **listen** begins listening on behalf of the service.

When service addresses are dynamically assigned, the assigned address is written to the **listen** log file.

listen and SAF

SAF provides a generic interface to which all port monitors must conform. **listen** is a port monitor under SAF's controller, SAC. (See "Overview of the Service Access Facility," "Port monitor management," and "Service management" for a description of SAF, the administrative files it maintains, and the commands used for port monitor and service administration.)

There can be multiple invocations of **listen** port monitors, each identified by a unique *pmtag*. Each of these port monitors can monitor multiple ports for incoming connection requests.

A port has one and only one service associated with it. Each port, and its associated service, is identified by a service tag, *svctag*. Service tags for any given port monitor are unique.

When SAC starts a port monitor, the port monitor reads its administrative file, which contains information about which ports to monitor and what service (that is, process) is associated with each port.

The nlsadmin command

SAF requires each type of port monitor to provide an administrative command. This command must format information derived from command-line options so that it is suitable for inclusion in the administrative files for that port monitor type. The command may also perform other port monitor functions.

nlsadmin is **listen**'s administrative command. The **nlsadmin** command formats information based on the options with which it is invoked and writes this information to the standard output.

nlsadmin is one of the arguments **pmadm** uses with the **-a** option to format information in a way suitable for inclusion in a **listen** administrative file. **nlsadmin** presents this information (as standard output) to **pmadm**, which places it in the file. This use of **nlsadmin** is described below under "Adding a listen port monitor." Port monitor specific information in a port monitor administrative file will be different for different port monitor types.

nlsadmin is also included on the **sacadm** command line when a port monitor is added to the system. It is used to supply the **listen** version number for inclusion in a port monitor's administrative file. The port monitor administrative file is updated by the Service Access Controller's administrative commands, **sacadm** and **pmadm**. **nlsadmin** merely provides a means of presenting formatted port monitor-specific data to these commands.

The **sacadm** command line uses **nlsadmin** only with the **-V** option. **nlsadmin -V** tells SAC the version number of the **listen** command being used.

Under SAF, it is possible to have multiple instances of **listen** on a single *net_spec*. A new option, **-N** *pmtag*, can be used in place of the *net_spec* argument. This argument specifies the tag by which an instance of **listen** is identified by the SAF. If the **-N** option is not specified (i.e. the *net_spec* is specified in the invocation), then it will be assumed that the last component of the *net_spec* represents the tag of **listen** for which the operation is destined.

Managing listen ports

This section discusses and includes examples of how to manage listen ports, port monitors, and services. It includes the syntax of relevant commands.

Listing configured listen port monitors

```
# sacadm  -l  [ -t listen ]
```

The **sacadm** command with only a **-l** option lists all port monitors currently defined for the system. For example:

```
PMTAG    PMTYPE  FLGS RCNT STATUS   COMMAND
tcp      listen  -    3    ENABLED /usr/lib/saf/listen tcp #listener for
tcp
ttymon1 ttymon   -    0    ENABLED /usr/lib/saf/ttymon #
```

Listing services configured for a listen port monitor

pmadm -l [**-p** *net_spec*] [**-s** *svctag*]

pmadm with only a **-l** will list all services for all port monitors on the system. If a port monitor is specified (**-p**), all services for that port monitor will be listed. The following is a sample listing for the command.

```
# pmadm -l -p tcp ↵

PMTAG PMTYPE SVCTAG FLGS ID      PMSPECIFIC
tcp   listen 101    -    listen - c - /usr/lib/uucp/uucico -r0 -unuucp -iTLI \
                                      #UUCP access direct to server
tcp   listen 102    -    listen - c - /usr/tcp/lib/ttysrv -d -n ntty,tirdwr,1d0 \
                                      #UUCP access to server via login
tcp   listen 1000   -    listen - c - /usr/tcp/lib/tstserver #TP TEST SERVER
tcp   listen 0      -    root   - c - sfbul.serve c - /usr/lib/saf/nlps_server \
                                      #NLPS server
```

The following command lines list the addresses associated with general **listen** service (**0**) or with login service (**1**):

pmadm -l -p *net_spec* **-s 0**
pmadm -l -p *net_spec* **-s 1**

By definition, service code **0** is for the **nlps_server**, which is a service that provides compatibility with pre-DG/UX Release 5.4 **listen** service requests. Service code **1** is for remote login (that is, **cu** over a network).

Adding a listen port monitor

sacadm -a -p *pmtag* **-t** *type* **-c** *cmd* **-v** `` `pmspecific -V` `` \
-n *count* [**-f dx**] [**-z** *script*] [**-y** *comment*]

The following example shows how **listen**'s administrative command, **nlsadmin**, can be used to obtain the current version number of **listen**'s administrative file when used with **sacadm** to add a **listen** port monitor.

```
# sacadm -a -p tcp -t listen -c "/usr/lib/saf/listen -m tcp" \
         -v `nlsadmin -V` ↵
```

This command line adds a line to SAC's administrative file. The options that may be used with **sacadm -a** are described under "Port monitor management" and in the **sacadm**(1M) and **nlsadmin**(1M) manual pages. If the port monitor being added has the same name as an existing port monitor, you must remove the old one before adding the new one.

Removing a listen port monitor

sacadm -r -p *net_spec*

For example:

```
# sacadm -r -p tcp ↵
```

SAC removes the line for port monitor **tcp** from its administrative file. The port monitor directory will remain in **/etc/saf** but will be removed and recreated when a new port monitor with the same name is added. To make changes to a port monitor entry, always remove the entry and add a new entry using the **sacadm** command. You are advised against editing the SAC administrative file.

Adding a service

```
pmadm -a -p { net_spec | pmtag } -s svctag -i id -m "`nlsadmin options `"
\        -v `nlsadmin -V` -y comment
```

The following example adds a new service, **/usr/bin/cmd**, to a **listen** port monitor whose tag is **listen**. The new service has service tag **23**, identity **guest**, and no private address:

```
# pmadm -a -p listen -s 23 -i guest -m `/usr/sbin/nlsadmin \
     -c /usr/bin/cmd` -v `/usr/sbin/nlsadmin -V` ↵
```

The same address cannot be monitored by more than one **listen** port monitor at any given time. The first attempt to listen on an address will bind successfully; subsequent attempts will fail to bind. If both static and dynamic addresses are monitored by more than one **listen** port monitor, the static addresses are bound first, then the dynamic addresses.

Mixing multiple **listen** port monitors — each of which has static and dynamic addresses specified — may result in unpredictable behavior. See "Adding a service" under "Service management" or the **pmadm**(1M) and **nlsadmin**(1M) manual pages for a full description of the **pmadm** command line options.

Removing a service

```
pmadm -r -p { net_spec | pmtag } -s svctag
```

For example:

```
# pmadm -r -p tcp -s 23 ↵
```

removes service **23** from the **tcp listen** port monitor.

Enabling and disabling services

```
pmadm -e -p net_spec -s svctag
pmadm -d -p net_spec -s svctag
```

To enable a service on a specific port, first find out which port monitor is monitoring the port. Enter:

```
# pmadm -l -t listen ↵
```

This lists all services defined for listen-type ports.

If the port monitor is **tcp** and the service tag is **101**, the command:

```
# pmadm -e -p tcp -s 101 ↵
```

will enable service **101**. To verify that the port has been enabled, enter

```
# pmadm -l -p tcp -s 101 ↵
```

The **x** will have been removed from the FLGS column in the entry for this service. When a service is disabled, all subsequent connection requests for the service will be denied. Using the same example:

```
# pmadm -d -p tcp -s 101 ↵
```

will restore the **x** to the FLGS field in the entry for service 101.

Disabling all services monitored by a listen port monitor

sacadm -d -p *pmtag*

To disable all services defined for the port monitor **tcp**, enter:

```
# sacadm -d -p tcp ↵
```

Any future connection requests for services managed by this port monitor will be denied until the port monitor is enabled. The command:

```
# sacadm -e -p tcp ↵
```

re-enables port monitor **tcp**.

Per-service configuration scripts

As a port monitor under SAF, **listen** can customize the environment of each service it starts. It does this by interpreting a per-service configuration script, if one exists, immediately before starting the

service. Per-service configuration scripts are optional. You install configuration scripts using the **pmadm** command with **-g** and **-z** options (see the **pmadm**(1M) manual page).

It is also possible to customize the environment of a **listen** port monitor. A per-port monitor configuration script is defined using the **sacadm** command with **-g** and **-z** options (see the **sacadm**(1M) manual page). The environment modifications made by a port-monitor configuration script are inherited by the port monitor and all the services it invokes. The environment of any particular service can then be customized further by using a per-service configuration script.

The **doconfig**(3N) manual page describes the language in which configuration scripts are written.

Configuration scripts are not normally needed for basic operations.

Log files

listen creates and manages the log files **/var/saf/***pmtag***/log** and **/var/saf/***pmtag***/o.log**. Log file entries are in the following format:

- *date time; PID; message*

where
date and *time* show when the entry was made.
PID is the ID of the process that made the log entry.
message gives a description of the event or error that caused the log message.

The following events are logged:

- Each connection that arrives
- Each service that is started
- Each file descriptor passed over a pipe
- State changes that occur
- Errors and unusual conditions

The log files are held open by **listen** processes. Entries are made by two types of processes: **listen** process (**listen**) and the NLPS server process (**nlps_server**). **nlps _server** is a service that provides compatibility with pre-DG/UX Release 5.4 service requests.

Port monitor management command reference

Command Syntax	Description
sacadm -a -p *pmtag* **-t** *type* **-c** *"cmd"* **-v** *ver* \ [**-f dx**] [**-n** *count*] [**-y** *"comment"*] \ [**-z** *script*]	Add a port monitor entry to SAC's administrative file.
sacadm -l [**-p** *pmtag* | **-t** *type]*	Print port monitor status information.
sacadm -L [**-p** *pmtag* | **-t** *type*]	Print port monitor status information in condensed format.
sacadm -G [**-z** *script*]	Print or replace per-system configuration script **/etc/saf/_sysconfig**.
sacadm -g -p *pmtag* [**-z** *script*]	Print or replace per-port monitor configuration script **/etc/saf/**pmtag**/_config**.
sacadm -e -p *pmtag*	Enable port monitor pmtag.
sacadm -d -p *pmtag*	Disable port monitor pmtag.
sacadm -s -p *pmtag*	Start port monitor pmtag.
sacadm -k -p *pmtag*	Kill port monitor pmtag.
sacadm -r -p *pmtag*	Remove the entry for port monitor pmtag from the SAC administrative file.

Service administration command reference

Command Syntax	Description
pmadm -a [**-p** *pmtag* | **-t** *type*] \ **-s** *svctag* **-i** *id* **-m** *"pmspecific"* \ **-v** *ver* [**-f ux**] [**-y** "comment"] \ [**-z** *script*]	Add a service entry to the port monitor administrative file.
pmadm -l [**-t** *type* | **-p** *pmtag*] \ [**-s** *svctag*]	Print service status information.
pmadm -L [**-t** *type* | **-p** *pmtag*] \ [**-s** *svctag*]	Print service status information in condensed format.
pmadm -g -p *pmtag* **-s** *svctag* \ [**-z** *script*]	Print, install, or replace per–service configuration script for service *svctag* associated with port monitor *pmtag*.
pmadm -g -s *svctag* **-t** *type* **-z** script	Install, or replace per-service configuration scripts for all services *svctag* associated with port monitors of type *type*.
pmadm -e -p *pmtag* **-s** *svctag*	Enable service *svctag* associated with port monitor *pmtag*.
pmadm -d -p *pmtag* **-s** *svctag*	Disable service *svctag* associated with port monitor *pmtag*.
pmadm -r -p *pmtag* **-s** *svctag*	Remove the entry for service *svctag* from the port monitor administrative file.

ttymon and terminal line setting command reference

Command Syntax	Description
sacadm -l [**-t** *type* | **-p** *pmtag*]	Lists all port monitors (**-l** alone), all port monitors of a given type (**-t** *type*), or a single port monitor (**-p** *pmtag*).
pmadm -l [**-t** *type* | **-p** *pmtag*] \ [**-s** *svctag*]	Lists all services for all port monitors (**-l** alone), all services for all port monitors of a given type (**-t** *type*), all services for a specific port monitor (**-p** *pmtag*), or a single service (**-s** *svctag*).
sacadm -a -p *pmtag* **-t ttymon** \ **-c** *cmd* **-v 'ttyadm -V'**	Adds a `ttymon` port monitor. **ttyadm** used with **sacadm -a** or **pmadm -a** as an argument to the **-v** option provides the comment line containing the **ttymon** version number for the new port monitor administrative file.
sacadm -r -p *pmtag*	Removes a port monitor.
pmadm -a -p *pmtag* **-s** *svctag* \ **-i** *id* [**-f ux**] **-v 'ttyadm -V'** \ **-m "'ttyadm** [**-b**] [**-r** *count*] \ [**-c**] [**-h**] [**-i** *msg*] [**-m** *modules*] \ [**-p** *prompt*] [**-t** *time-out*] \ **-d** *device* **-l** *ttylabel* \ **-s** *service*'"	Adds a service. **ttyadm** used with **pmadm -a** as an argument to the **-m** option provides the *pmspecific* fields for inclusion in the port monitor's administrative file.
pmadm -r -p *pmtag* **-s** *svctag*	Removes a service.
pmadm -e -p *pmtag* **-s** *svctag*	Enables a service.
pmadm -d -p *pmtag* **-s** *svctag*	Disables the service *svctag*, available through port monitor *pmtag*.
sacadm -e -p *pmtag*	Enables all services defined for port monitor *pmtag*.
sacadm -d -p *pmtag*	Disables all services defined for port monitor *pmtag*.
/usr/sbin/sttydefs -a *ttylabel* \ [**-b**] [**-n** *nextlabel*] \ [**-i** *initial-flags*] \ [**-f** *final-flags*]	Adds an entry to the **/etc/ttydefs** file.
/usr/sbin/sttydefs -l [*ttylabel*]	Prints terminal line setting information from the **/etc/ttydefs** file for terminal ports with the label *ttylabel*. If no *ttylabel* is specified, prints terminal line setting information for all records in the file.
/usr/sbin/sttydefs -r *ttylabel*	Removes records for the *ttylabel* specified from **/etc/ttydefs**.

listen command reference

Command Syntax	Description
sacadm -l [**-t** *type*]	Lists status information for all port monitors (**-l** alone) or for all port monitors of a given type (**-t** *type*).
pmadm -l -p *net_spec* [**-s** *svctag*]	If *svctag* is supplied, lists status information for the service. If no service is specified, lists status information for all services under *net_spec*.
sacadm -a -p *net_spec* | *pmtag* \ **-t listen** \ **-c "/usr/lib/saf/listen** *net_spec*" \ **-v 'nlsadmin -V'**	Adds a **listen** port monitor.
sacadm -r -p *net_spec*	Removes a **listen** port monitor.
pmadm -a -p *net_spec* | *pmtag* \ **-s** *svctag* **-i** *id* \ **-m "'nlsadmin** *options*'" \ **-v 'nlsadmin -V' -y** *comment*	Adds a service under a **listen** port monitor.
pmadm -r -p *net_spec* | *pmtag* \ **-s** *svctag*	Removes a service under a **listen** port monitor.
pmadm -e -p *net_spec* **-s** *svctag*	Enables service *svctag* under port monitor *net_spec*.
pmadm -d -p *net_spec* **-s** *svctag*	Disables service *svctag* under port monitor *net_spec*.
sacadm -d -p *net_spec*	Disables all services under port monitor *net_spec*.
sacadm -e -p *net_spec*	Enables all services under *net_spec*.

End of Chapter

12 Controllers

This chapter tells how to manage the following kinds of controllers:

- Synchronous VSC controllers, used for wide-area networks (WANs)
- VLC controllers, used for local-area networks (LANs)
- Asynchronous VDA controllers, used for terminal lines and other asynchronous connections
- Virtual terminal controllers (VTCs), used to allow Ethernet devices running TCP/IP to connect to an AViiON host

Managing synchronous WAN controllers

This section describes how to manage your system's synchronous wide-area network (WAN) controllers. The supported sync controllers include the VSC/3, VSC/3i, and VSC/4 (VME Bus Synchronous Controller) controllers. The **sysadm** Device–> Sync menu contains operations for starting, stopping, checking, and listing your sync controllers.

You can use the Start, Stop, Check, and List operations only if intelligent synchronous communications drivers are configured on your system. If the controllers are installed on the system when you build your kernel, the system file will include the correct driver entries. The VSC/3 and VSC/4 controllers require the **ssid** driver, and the VSC/3i controller requires the **vsxb** driver.

The **sysadm** `Device-> Sync` operations are appropriate only for upper layer communications software such as X.25 and SNA. These operations are not appropriate for integrated synchronous controllers such as **iscd** and **izscd**.

Starting sync controllers

Use the **sysadm** operation `Device-> Sync-> Start` to download the controller-resident software to sync drivers and initialize them. The operation prompts you for the controllers that you wish to start. Select **all** to start all controllers.

You may not perform this operation on a controller if any port on the controller is in use.

Stopping sync controllers

Use the **sysadm** operation `Device-> Sync-> Stop` to halt all controller-resident software running on a sync controller. The

operation stops the controller by performing a hardware reset on the controller board. The operation prompts you for the sync controllers that you wish to stop. Select **all** to stop all controllers.

You may not perform this operation on a controller if any port on the controller is in use.

Checking sync controllers

Use the **sysadm** operation `Device-> Sync-> Check` to verify that sync controllers are functional. The operation prompts you for the controllers that you wish to check. Select **all** to check all controllers.

This operation checks controllers by calling the **synccheck**(1M) command. The **synccheck** command tests a controller by verifying that controller-resident software has been downloaded and that the controller can perform DMA operations across the VME bus.

Listing sync controllers

Use the **sysadm** operation `Device-> Sync-> List` to display the **/dev** entries for downloadable sync controllers configured on your system. The display may include entries such as **ssid** and **vsxb**.

Managing LAN controllers

This section describes how to manage local-area network (LAN) controllers that execute controller-resident software downloaded from the host and list all LAN controllers. The **sysadm** `Device-> LAN` menu contains operations for starting, stopping, and listing your LAN controllers.

You can use the start and stop operations only if intelligent LAN communications drivers are configured on your system. If the controllers are installed on the system when you build your kernel, the system file will include the correct driver entries. The VLCi controller is intelligent and requires the **cien** driver.

Starting LAN controllers

Use the operation `Device-> LAN-> Start` to download controller-resident code to the LAN controller and initialize it. The operation prompts you for the LAN controllers that you wish to start. Select **all** to start all controllers.

Stopping LAN controllers

Use the operation `Device-> LAN-> Stop` to halt all tasks running on a LAN controller. The operation stops the controller by

performing a hardware reset on the controller board. The operation prompts you for the LAN controllers that you wish to stop. Select **all** to stop all controllers.

Listing LAN controllers

Use the operation `Device-> LAN-> List` to display the **/dev** entries for your LAN controllers.

Managing VDA controllers

This section describes how to manage your VDA (**syac**) controllers. VDA controllers handle the asynchronous terminal lines on your system. There are no **sysadm** operations for managing VDA controllers. Instead, you use the **tcload**(1M) command.

To start specific asynchronous terminal controllers, specify the device node from the **/dev/async** directory. For example:

```
# tcload  "/dev/async/syac@60(60000000)" ↵
```

To start all asynchronous terminal controllers, use this command line:

```
# tcload  -a ↵
```

To reset a specific VDC cluster box without resetting all lines on the associated VDA controller, turn the power to the cluster box off and then on. When power returns to the cluster box, the system downloads the required code to the cluster box without user intervention.

Managing virtual terminal controllers (VTCs)

The virtual terminal controller, also called a VTC, allows Ethernet devices running TCP/IP to connect to an AViiON host and appear as directly connected terminals. It does this by running the entire TCP/IP protocol stack on the board. This TCP/IP implementation is completely separate from the DG/UX TCP/IP implementation. The VTC is managed differently than DG/UX TCP/IP. The VTC can only be used for emulation of asynchronous device access to the host. In a typical environment, this means users may need to use two different names (mapped to two different Internet addresses) to access the same AViiON host.

The interface the board presents to the AViiON host is the same as that of other asynchronous connections. This means that programs running over the VTC will not know they have a networked connection. Users will see their console as a tty, not as a ttyp, for example.

This section is a guide to the steps you need to take to configure the DG/UX system, the VTC, and your terminal server to make the VTC operate properly.

Configuring the DG/UX kernel to include VTCs

You modify the system kernel by going into **sysadm** System–> Kernel–> Build menu and adding the necessary line for each VTC. The kernel must be built and the system rebooted for the changes to take effect. Each VTC in the system must be included in the kernel. The VTC, VAC/16, and VDA/255 use the same device names. In your system, syac(0) might be a VDA/255 and syac(1) might be a VTC.

Each device must appear on a separate line of the form:

```
syac(n)          ## Systech terminal controller
```

The actual jumpering of the board determines what number should be in place of *n* in `syac(n)`. The following is an example of a kernel device specification for a system with five VTCs installed:

```
lp()              ## Integrated parallel line printer controller
duart(0)          ## Dual-line terminal controller (number 0)
hken()            ## Hawk Ethernet controller
syac(0)           ## Systech terminal controller (number 1)
syac(1)           ## Systech terminal controller (number 2)
syac(2)           ## Systech terminal controller (number 3)
syac(3)           ## Systech terminal controller (number 4)
syac(4)           ## Systech terminal controller (number 5)
sd(insc(),0)      ## SCSI disk 0 on Integrated SCSI adapter
st(insc(),4)      ## SCSI tape 4 on Integrated SCSI adapter
```

DG/UX assigns 256 ttys to each VTC configured into the system. It is important to note that tty assignments are based on the order of devices in the system configuration. In the example above, `duart(0)` will allocate and control tty00. The next device requiring ttys is `syac(0)` and it will control the next 256 ttys, tty01 through tty256. This is an important concept when assigning Internet

addresses to specific ttys. There is a one-to-one mapping between the tty name and the logical channel on the VTC.

Assigning IP addresses and ports to VTCs and ttys

The VTC supports both incoming and outgoing network connections. Each logical channel on the VTC can be configured for either an incoming network connection or an outgoing network connection. Specific Internet addresses, TCP/IP port numbers, and protocols can be specified as well.

Each VTC in your system must be assigned an Internet address. The default behavior is that all of the ttys on the board will respond to an incoming connection to that Internet address, listening on TCP/IP port 23, and running the **telnet** protocol.

For all incoming connections, the 256 logical channels on the board are searched in low to high order, looking for an open unused channel whose Internet address and port matches the incoming request. That channel will be used for the duration of the session. Because connections come in randomly, users may be matched to different channels and will see different tty names each time they log on the system.

Some programs require a user to connect to the same tty for each network connection. There are two ways of doing that. One way is to assign a specific Internet address which is different from the global board address to the tty. The terminal server must then connect to that specific Internet address. Another way is to specify a unique tcp port number for the tty. The tty will respond to the board's Internet address, but only when addressed by that unique port number. A combination of the two forms may be used.

To provide compatibility with some other terminal servers, the VTC supports two additional access protocols, raw tcp access and **rlogin**. With raw tcp access, the TCP/IP connection is used as a transparent 8-bit data pipe between the VTC and the other device. Incoming connections can be made using the **rlogin** protocol as well. Although the VTC supports the **rlogin** protocol, the user does not get logged in automatically and has to re-enter the username/password pair.

The VTC supports automatic establishment of outgoing connections using the **telnet** and raw TCP/IP protocols. If the channel is specified as callout, a connection request is initiated on the network when the tty is opened by DG/UX. If unsuccessful, the VTC will automatically retry every eight seconds while the tty stays open. If the tty is closed by DG/UX, the VTC will stop attempting the connection. If the connection is opened successfully but

subsequently broken, the user must close then reopen the tty for the VTC to attempt to re-establish the connection.

The file **/etc/tcload/vtc.addrs** contains configuration information for VTC boards. The information is communicated to the board during the download process. There must be an entry in this file for each VTC specifying the default Internet address, broadcast address, netmask, and the routing information the board should use. The file may also contain entries for any ttys that require a nonstandard configuration. See the **vtc.addrs** man page for information.

The default routing specification can be used if the VTC is attached to the same Ethernet as the DG/UX LAN controller. If default routing is specified, the VTC will use the same routing information that exists in the host when the board is loaded. However, at system initialization time, no routing information may exist. Once DG/UX TCP/IP is running, the **syac_routes** command can be used to load the current routing information.

If the VTC is going to be on a different network than the DG/UX LAN controller, you must specify a file containing the routing information. This file has the same format as the **/etc/gateways** file. If no routing is desired, this file may be empty, but it must still be specified and present on the system.

The general form for specifying tty specific information is:

```
/dev/ttyxx  internet_address@port#  protocol-direction
```

The *internet_address* is one of three forms:

128.221.222.222@6400
    128.221.222.222 is the internet address
    6400 is the port

128.221.222.222
    no port number specified

@6400
    Internet address the same as the board

The *protocol–direction* is one of five possibilities:

telnetin – Accept an incoming **telnet** connection.
    This is the default behavior.
    If only the Internet address is used, the port is assumed to be 23. If only the port number is used, the board Internet address is assumed.

rlogin – Accept an incoming **rlogin** connection.
If only the Internet address is used, the port is assumed to be 513. If only the port number is used, the board Internet address is assumed.

tcpin – Accept an incoming raw tcp connection.
The Internet address-only form is not allowed. If only the port number is used, the board Internet address is assumed.

telnetout – Make a **telnet** connection out.
If only the Internet address is used, the port is assumed to be 23. The port number-only form is not allowed.

tcpout – Make a raw connection out.
The Internet address and port must both be specified.

The VTC has no way of ensuring the protocol it is running for a channel is the same as the remote device. You should take care that both network devices run the same protocol. If they do not, errors can occur. For example, if the VTC is configured to run raw TCP/IP and the terminal server thinks it is running **telnet**, then the **telnet** negotiation sequences will be assumed to be data by the VTC and passed to the application program.

The DG terminal server product automatically maps TCP/IP ports to channels in a linear fashion and makes assumptions about the protocol to be run on the basis of the port number. If you assign ports and protocols in the same way, no protocol mismatch should occur.

Ports 6400 to 6655 map to channels 0 through 255 running the **telnet** protocol. Incoming connections that are rejected will be opened momentarily, a text message will be sent on the connection, and then the connection will be closed.

Ports 6912 to 7167 map to channels 0 through 255 running the raw TCP/IP protocol. Incoming connections that are rejected will be opened momentarily, a text message will be sent on the connection, and then the connection will be closed.

Ports 7424 to 7629 map to channels 0 through 255 running the **telnet** protocol. Incoming connections that are rejected will be issued a TCP **rst** packet.

Ports 7936 to 8191 map to channels 0 through 255 running the raw TCP/IP protocol. Incoming connections that are rejected will be issued a TCP **rst** packet.

Reloading a VTC

The **tcload** command reloads a terminal controller. With a VTC, **tcload** resets the board, downloads its resident controller code, loads the Internet addresses, and begins normal execution. All sessions previously running on the VTC are terminated when the load occurs. See the **tcload** man page for more information about this command.

Changing VTC routing information

The **vtc_routes** command changes the routing information used by the specified VTC. The old routing information is normally flushed from the board. The new routing information is taken from the file specified in the **vtc.addrs** configuration file for the device. If the routing specification is set to the default, the current routing information from DG/UX will be read and automatically loaded in the VTC. See the **vtc_routes** man page for the format of this command.

Setting VTC tty-specific information

The **vtc_ttyaddrs** command allows you to set tty-specific information at any time. The syac specified must be a VTC. The **vtc_ttyaddrs** man page has information about the format of this command.

You can also specify tty-specific information in the file **/etc/tcload/vtc.addrs**. When the information is specified in this file, it is loaded automatically during system boot or controller reload.

Adding terminals on the VTC

DG/UX ttys for the VTC must be monitored by a port monitor in order to be accessible. You use **sysadm** to add ttys on the VTC to a port monitor: Go into the **sysadm** Devices–> Port–> Terminal–> Add menu and enter the tty definition label as `M9600`. We recommend that there be no more than 64 ttys assigned to any one port monitor.

Some ttys, especially modems and printers, require a modification of the port services to accommodate features such as bi-directional (**b**), connect-on-carrier (**c**), do not hang-up on last close (**h**), and so on. To make modifications to the port services for a specific tty, use the **sysadm** Devices–> Port–> Port Services menu.

Enabling user login and logoff via a VTC

When a user connects to an AViiON host via a VTC, a login banner will not appear until a newline is struck. If you want the login banner to appear automatically, set the port services option connect-on-carrier (**c**) must be set for that line using the **sysadm** Devices–> Port–> Port Services menu.

By default, the network connection is brought down when a user logs off DG/UX via a VTC. This can be changed with the **stty** option **–hupcl**. If **–hupcl** is set, no network hangup will occur on the last close; the network connection is put into a holding state until the user process is back up. For this to operate with **ttymon**, the port services option hangup (**h**) must be set to **NO** and the configuration variable CHECK_IN_USE for the controlling ttymon must be set to 0. For information on the CHECK_IN_USE configuration variable, see the **ttymon**(1M) man page.

Configuring printers with a VTC

Printers that are connected to a port on a terminal server can be set up as system printers on an AViiON host with a VTC.

The printer must be added using the **sysadm** Devices–> Printer–> Devices–> Add menu. The printer should be added to the system exactly as though it were connected directly, with two exceptions.

The first exception is that you should specify **–hupcl** as an **stty** option in the printer setup to keep the connection open between the VTC and the terminal server between printer requests.

The second exception is that the terminal server port should then be configured. A named configuration file on Data General's TermServer product, **dgprinter**, can be used to define the default parameters of the port to which the printer is attached.

The network configuration can be set up one of two ways. You can have the terminal server automatically issue the connection to the VTC. In that case, you should include an entry in the **vtc.addrs** file assigning an Internet address/port number to the printer tty. The printer port on the terminal server must be set up with a permanent virtual circuit that will automatically connect to the Internet address/port number associated with the tty for the printer that is specified in the **vtc.addrs** file.

Alternatively, the VTC could automatically issue the connection to the terminal server. The tty entry would specify the Internet address and port number for connection to the terminal server as well as the appropriate protocol. If you are using the Data General TermServer product, we suggest you use the port number range that assumes raw TCP/IP access with a TCP **rst** message on connection rejection.

Connecting a modem to a host with a VTC

Modems that are connected to a port on a terminal server may be used as connections to an AViiON host via a VTC for CU or UUCP access. This network connection should be configured one of two ways just as described for setting up printers.

You must select one of the ttys on the VTC for use by UUCP. The port services option bi-directional (**b**) must be set as described in "Adding terminals on the VTC." Also, you must include an entry in the **vtc.addrs** file for the tty that UUCP will be using.

If you do not want the echo during login, you must modify the **/etc/ttydefs** file. Locate the line that starts with the modem speed you are using — for example, `M2400` — and insert a dash (**-**) before the first **echo**, **echoe**, and **echok** parameters. These first entries control echo during login only. Do not put the dash before the second echo entries on this line; the second echo entries control the post login session.

IMPORTANT: Do not put **,M** after the tty entry in the Devices file, as you may have done for directly connected autodialing modems. The terminal server handles the physical modem connection and handshaking, not the host.

You must configure the terminal server port for the modem. A named configuration file on Data General's TermServer product, **dghost**, may be used to define the default parameters of the port to which the modem is attached. You must also modify the parameters to match the flow control algorithm the modem is expecting. The **dcd** parameter should generally be set to **onconnection**; with the standard TermServer-to-modem cable, this allows the terminal server to reset the modem between each connection. The **dtr** parameter should generally be set to **asdcd**; this permits the terminal server to send data to the modem even if carrier detect is not asserted. Finally, you must specify the permanent virtual circuit to the Internet address associated with the tty.

The following instructions are based on a Hayes-compatible asynchronous modem. If you are using a different type of modem, you will need to refer to the owner's manual for the modem for the appropriate settings.

Set the modem switches as follows:

Front: Switch 2 down – all others up
Rear: All up

You need to modify the default factory settings for the modem by connecting to the modem with a terminal and entering the following command:

```
at&fs0=1&d3&c1\d1q1&w<cr>
```

This command does the following:

at&f	Returns the modem to factory settings.
s0=1	Sets the modem to auto-answer on the first ring.
&d3	Lets the terminal server control the modem via DTR.
&c1	States that CD follows modem carrier.
\d1	DSR follows off-hook, and CTS follows CD.
q1	Disables output of result codes.
&w	Writes the changes.

If you are using a VTC modem with any of the UUCP applications such as **uucp**, **cu**, or **uucico**, the modem is be handled as if it were directly connected with only a few considerations. These may also apply to other applications that use these modems.

The chat script that initiates the call should include the **Q0** command to enable the output of result codes for the outgoing call. The modem will revert to quiet mode after the call is disconnected in preparation for an incoming call.

Other documentation related to the VTC

The following hardware documentation is related to installing and setting up the VTC:

- *VMEbus TermServer Controller (VTC) Hardware Technical Manual* (014–002108)
- *Technical Notice: Setting Up and Installing VMEbus TermServer Controllers (VTC)* (014–002109)

The man pages for **vtc.addrs**, **vtc_ttyaddrs**, **vtc_routes**, and **syac** provide information about specific commands and formats.

End of Chapter

13 Loading and setting up application software

This chapter outlines the procedure for loading and setting up additional software packages (such as TCP/IP or X windows) you received with the DG/UX system.

The software packages on your system fall into the following categories:

- Software packages that conform to DG/UX system load and setup guidelines.
- Software packages that conform to 88open Consortium load and setup guidelines.
- Software packages that provide their own customized **sysadm** menus and operations.

Adding a package (obtained from either Data General or a third-party vendor) requires two steps: loading from the release tape, and setting up the package to run with the DG/UX system. If you received a Data General release tape and postponed the setup of any of the bundled packages (such as the DG/UX X Window system, TCP/IP, ONC, or NFS), you can follow the procedures in this chapter. However, you will need to supply some answers to prompts about TCP/IP and ONC during the **sysadm** setup dialogue. Refer to the installation planning worksheets that you completed in *Installing the DG/UX™ System* for this information.

Handling packages conforming to DG/UX standards

The Package menu provides operations for handling software packages shipped by Data General for installation on DG/UX systems. The Data General ONC™/NFS® network software package and the X Window System software package are examples of Data General packages that you might load with these operations. Packages from other sources may also conform to the DG/UX package guidelines, which are discussed in *Porting and Developing Applications on the DG/UX™ System*.

Before you load any software package, you should consult the package's release notice. The release notice should contain vital information such as prerequisite software packages, where the package will load, and how much disk space it will require.

Typically, optional DG/UX packages load into a directory in **/usr/opt**. This directory should actually be the mount point for a

virtual disk and file system created exclusively for holding the package. The reason for creating a dedicated file system like this is to avoid over-filling the **/usr** file system and to make future upgrades of the DG/UX system (which may involve overwriting **/usr**) less complicated.

All packages require that you load them, and many also require that you set them up. Loading involves simply transferring the software files from the distribution media to disk. Setting up a package involves running scripts provided with the package. These scripts may execute without requiring user interaction, or they may require that you supply some information necessary to initialize the software. Loading and setting up are both generally quite easy because the typical DG/UX package contains scripts to automate the procedures.

The **sysadm** Software–> Package menu provides operations for installing, loading, setting up, and listing DG/UX packages.

Installing software into a release area

Install software packages with the **sysadm** operation `Software-> Package-> Install`.

This operation is intended for packages that conform to the DG/UX package guidelines as specified in *Porting and Developing Applications on the DG/UX™ System*.

Installation involves loading and setting up the software. Loading the software means transferring it from the media, such as tape, to disk. Setting up the software means running the setup script provided with the package to perform tasks such as initializing databases or querying you for needed information, for example. The **sysadm** utility handles loading, and scripts provided with the software may handle the setup procedures.

During the setup phase of package installation, the Install operation executes only the setup script having the same name as the package, if any. For example, if you install a package named Officeware that includes setup scripts named Officeware, Spreadsheet, and Wordprocess, the Install operation will run only the Officeware setup script. The operation does not run any other setup scripts; rather, at the end of the operation, it notifies you that other scripts exist. To execute them, select the Setup operation.

Before installing any software package, consult the package's release notice. The release notice contains vital information about prerequisite software, the location on disk where the software will reside, the amount of disk space that the package requires, and so

forth. The release notice should tell how to install the package. The instructions in this manual are general and may not apply completely to the package that you are loading.

Before installing a software package, you need to know into which release area to load it. A release area is a directory structure intended to contain operating system software to support OS clients. Every system has a primary release area, which contains the currently installed release of the DG/UX system. If you are installing the package for use by OS clients that do not use the primary release, see "Installing for OS clients."

A caution about disk space

Be careful not to load a package into a file system that does not have enough free space. To verify how much space a file system has, use the **df**(1M) command. The following example shows how to list the amount of free space in your **/usr** file system.

```
% df -t /usr ↵
/usr  (/dev/dsk/usr  ):   22610 blocks   27253 files
                 total:  240000 blocks   34560 files
```

The **df** output above shows that the **/usr** file system has 22,610 512-byte blocks of free space, or approximately 11.5 Mbytes.

If the file system does not have enough space, expand the file system with the **sysadm** operation `File System-> Local Filesys-> Expand`. Remember that you cannot boot from a file system built on an aggregated virtual disk that spans multiple physical disks; therefore, do not expand the / (root) or **/usr** file systems if doing so would add another partition on a separate physical disk. If you need to add space to / or **/usr**, first make sure that the desired amount of free space exists on the same physical disk.

Remember that for whatever amount of physical disk space you allocate, approximately 7% will be used for file system overhead, and 10% will be the free space buffer. Thus, you should allocate 17% more disk space than you intend to use. For example, if you need to allocate space for a 50 Mbyte software package, allocate 59 Mbytes of disk space.

If you attempt to load the package into a file system that does not have enough space, the operation will fail. Depending on which file system it is that becomes full, you may also disrupt any current users' access to the file system. The disruption can be particularly inconvenient if you fill up the / or **/usr** file systems. If the operation fails, it will leave the software package partially loaded on disk, and

you will have to remove the loaded files yourself. To find out what files to remove, see the release notice that shipped with the package.

For how to expand file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Installing for OS clients

If you are installing software for OS clients (diskless workstations) that use an operating system release other than the primary release, you will have to supply the appropriate release name when you install the software. The release area must already exist. If your OS clients use the same release of the DG/UX system that your OS server uses, you may install the package in the primary release area.

The installation operation loads any / file system files not only into your system's / file system but also into the OS client prototype / file system (which appears on the OS server as **/usr/root.proto**). Loading a software package on your system provides the option of loading it into the prototype / file system as well.

The prototype / file system is what **sysadm** uses as the model / file system when creating a new OS client root area. The operation you use to add an OS client copies the prototype root when making the initial root file system for the new OS client. This way, the OS client gets a copy of the operating system software as well as any other loaded packages.

Loading software into a release area

To load software from the media to disk without setting it up, you use the **sysadm** operation `Software-> Package-> Load`. For the specified release, this operation loads the software into the / and **/usr** file systems. The operation also loads the root portions of the package into the prototype root and any existing OS client roots so that systems with OS clients will also have the package.

Read the software release notice before loading to see where the software will load and how much disk space it requires. Make sure your system has enough free disk space in the appropriate areas. If you need to create additional file systems for the package, make sure they are mounted before loading the package.

If you create new file systems for a software package, OS clients who need access to the package will have to mount the new file systems onto the appropriate directory on their system.

These instructions assume that the package conforms to Data General package specifications. Most vendors' packages are loaded at **/usr/opt/***package-name*, where *package-name* is the name of the package. However, the package's release notice specifies the location.

IMPORTANT: If you are setting up only the DG/UX packages for an OS client, refer to Chapter 18 for instructions. Also, if you are loading or setting up the ONC/NFS package, you must declare your computer system an NIS master or server as defined in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

1. Before you load and set up a software package, read its release notice to find out where it will load and how much disk space it requires. You must create the required virtual disks and their file systems, and add and mount them on the DG/UX file system before you can load the package. For many packages, the installation script leads you through the disk, file system, and mounting phases. But if you need help with these tasks, see *Managing Mass Storage Devices and DG/UX™ File Systems*.
2. Insert the software medium, tape or CD-ROM, in its drive and make sure the drive is ready.
3. To load a package, follow this path through **sysadm**:

   ```
   Software-> Package-> Load
   ```

   If there is just one release area, **sysadm** prompts for the load device; skip to step 4. If there is more than one release area, **sysadm** prompts:

   ```
   Release Area: [PRIMARY]
   ```

   You have a choice about where to install the package, in the primary area or a secondary area. If you want to install it in a secondary area (perhaps for access by an OS client that has special needs not satisfied by the primary DG/UX release), you can do so. If **sysadm** is not displaying a list of release areas, enter ? to see the list of areas; then specify the release area you want. Chapter 14 explains adding a DG/UX release in a secondary area. In most cases, you will want the release in the primary area. For example:

   ```
   Release Area: [PRIMARY] ↵
   ```

4. **Sysadm** prompts:

   ```
   Release Medium: [/dev/rmt/0]
   ```

5. For a list of load devices, enter **?**; then specify the device (tape or CD-ROM drive) that holds the release medium. For example:

```
Release Medium: [/dev/rmt/0]  ? ↵
Choices are

     1  /dev/rmt/0
     2  /dev/rmt/1
     3  /dev/rmt/2
     4  /dev/pmt/0
     5  /dev/pmt/1
     6  /dev/pmt/3
     7  /dev/dsk3
Release Medium: [/dev/rmt/0] 2 ↵
Is /dev/xxx/n ready? [yes]
```

6. If the device is ready, press Enter; otherwise, answer **n**, make sure the device is ready, and press Enter. For example:

```
Is /dev/xxxn ready? [yes] ↵
Package Name(s): [all]
```

7. For a list of the packages on the release tape, enter ?:

```
Package Name(s): [all] ?↵
Choices are

1 all
2 fredware
3 fredware.man
4 fredware.rn
5 fredware.int
```

8. After reading the package's release notice, select the desired packages for loading. If you do not select all packages, specify the numbers or names of the individual packages. For multiple packages, type each package's name followed by a comma (,) or space. For example:

```
Package Names(s) [all] 2–4 ↵
```

or:

```
Package Names(s) [all] fredware, fredware.man ↵
```

To load all packages on the release tape, press Enter to accept the **all** default.

At this point for tape, the tape drive will advance and rewind; for CD-ROM, the drive will access the disk. The time required to load a package and the exact messages written to your screen depend on the particular package being loaded. A sample dialog for package **fredware** follows:

```
Package Name(s) [all] ↵
List file names while loading? [no] ↵
OK to perform operation? [yes] ↵

Positioning the tape to load: fredware .....
Loading the package: fredware ...
Loading the package: fredware.man  ......
Loading the package: fredware.rn ......
Loading the package: fredware.int ......

The package "fredware" has been loaded.
Package load is finished.
The selected packages have been loaded.
```

Setting up software in a release area

After loading packages, you use **sysadm** to set up the packages. Setup involves running scripts that accompany the package. A script initializes files and moves them to the appropriate locations in the DG/UX file system. Most packages are set up in **/usr/opt/***package*, where *package* is the name of the package. Some scripts require customizing; if so, the script will prompt you for answers. You may need to read the package's release notice for details.

Setting up software may consist of any number of steps, all determined by the setup script, if any, loaded with the package. The setting up process involves initializing any files as necessary and distributing files to their required locations on the system. The setup script often prompts you for information necessary to customize the software for your site. There may also be other steps you need to complete before the software package is usable; consult the package's release notice for more information.

To set up software that you have already loaded, you use the **sysadm** operation `Software-> Package-> Set up`. For a given release area, this operation sets up the software in both the **/usr** and root file systems. The operation also lets you set up the software in the roots of OS clients attached to the release. Whether or not you set the software up in OS clients' roots depends on the nature of the given package's setup script and whether or not the users of the OS clients prefer to set up their own systems. This operation locates all setup scripts that have not been run from a software package and allows the user to execute them.

The next three sections explain setup on a stand-alone computer system, an OS server, and an OS client.

Setting up software on a stand-alone computer

Use the steps in this section if your computer is neither a server or a client; that is, your system will run using the software on its own local disks and no other system will use that software.

1. To set up one or more packages, follow this path through **sysadm**:

   ```
   Software-> Package-> Set up
   ```

   If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the package name to be set up:

   ```
   Package Name(s):  [all]
   ```

2. Enter **?** for a list of packages that have been loaded but not yet set up. For example:

   ```
   Package Name(s):  [all] ? ↵

   Select the package(s) you want to set up.  If you
   want to set up all packages, select 'all.'  If you
   select 'all,' do not select any individual package
   names.

   Choices are

   1 all
   2 fredware
   3 fredware.man
   4 fredware.rn
   5 fredware.int

   Package Name(s) [all]
   ```

3. Specify the package names or press Enter for the default; for example:

   ```
   Package Name(s) [all] ↵
   OK to perform operation? [yes] ↵
   ```

4. If your answers are correct, confirm by pressing Enter:

```
OK to perform operation? [yes] ↵

Setting up fredware in usr.

Package fredware has been successfully set up in
usr.
Setting up fredware in MY_HOST root.
Package fredware has been successfully set up in
    MY_HOST root.
Package setup for fredware is complete.
```

The time required to set up the packages and the prompts depend on the package. Consult the package release notice for details.

Setting up software from an OS server for OS clients

You can set up packages at the OS server for the server and/or for OS clients. You can also set up packages for an OS client from that client. This section assumes you will set up packages from the OS server.

1. To set up one or more packages on a server, follow this path through **sysadm**:

```
Software-> Package-> Set up
```

If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the client names:

```
Client(s) to be set up: [none]
```

If you want to perform package setup for some or all OS clients, at the first prompt enter **?** for a list of OS clients. For example:

```
Client(s) to be set up: [none] ? ↵
```

```
Select 'all' to set up package(s) in the root file
system of all clients attached to the selected
release area.  Select individual clients if you want
to set up the package only for this client.

Select nothing or 'none' to set up the package only
in your own root file system and not in the root
file system of any clients.  Typically, clients set
up their own packages.

Choices are

1 all
2 none
3 bart
4 junior

Client(s) to be set up: [none]
```

2. To perform package setup for the OS server only, take the default, **none**. To perform package setup for one or more clients, enter their names, separated by commas, or enter **all** for all clients. For example:

```
Client(s) to be set up: [none] all ↵
Package Name(s):  [all]
```

3. To list of packages that have been loaded but not yet set up, enter **?** For example:

```
Package Name(s):  [all] ? ↵

Select the package(s) you want to set up.  If you
want to set up all packages, select 'all.'  If you
select 'all,' do not select any individual package
names.

Choices are

1 all
2 fredware
3 fredware.man
4 fredware.rn
5 fredware.int

Package Name(s):  [all]
```

4. Specify the package name(s) or press Enter for the default. For example:

```
Package Name(s) [all] ↵
OK to perform operation? [yes] ↵
```

5. If your answers are correct, confirm by pressing Enter:

```
OK to perform operation? [yes] ↵

Setting up fredware in usr.

Package fredware has been successfully set up in
usr.
Setting up fredware in MY_HOST root.
Package fredware has been successfully set up in
    MY_HOST root.
Package setup for fredware is complete.
```

The time required to set up the packages and the prompts depend on the package. Consult the package release notice for details.

Setting up software from an OS client for OS clients

You can set up an OS client package at either the OS server or at each OS client that is fully operational (the OS client has booted its kernel). This section assumes you will perform setup at the OS client after its kernel is booted. To perform setup for OS clients at the server, see the previous section.

1. To set up one or more packages on an OS client, follow this path through **sysadm**:

```
Software-> Package-> Set up
```

If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the package names:

```
Package Name(s):  [all] ? ↵
```

2. Enter **?** for a list of packages that have been loaded but not yet set up. For example:

```
Package Name(s):  [all] ? ↵

Select the package(s) you want to set up.  If you
want to set up all packages, select 'all.'  If you
select 'all,' do not select any individual package
names.

Choices are

1 all
2 fredware
3 fredware.man
4 fredware.rn
5 fredware.int

Package Name(s):  [all]
```

3. Specify the package names or press Enter for the default. For example:

```
Package Name(s) [all] ↵
OK to perform operation? [yes] ↵
```

4. If your answers are correct, confirm by pressing Enter:

```
OK to perform operation? [yes] ↵

Setting up fredware in usr.

Package fredware has been successfully set up in
usr.
Setting up fredware in MY_HOST root.
Package fredware has been successfully set up in
    MY_HOST root.
Package setup for fredware is complete.
```

The time required to set up the packages and the prompts depend on the package. Consult the package release notice for details.

Listing packages

To display information about software packages on release media or installed on your system, select the **sysadm** operation `Software-> Package-> List`.

When listing packages installed on a system that has multiple release areas, you need to specify which release area's packages to list. You may choose to list any or all of the packages that have been installed in that release area.

The default listing shows only the short name of each loaded package. For packages loaded on the system, the detailed listing shows the short name, the complete name, the release number of the package, and the date the package was created or released. For packages on release media, you see the table of contents, which includes the package's **tar** or **tarZ** components, component sizes, component load points, and so on.

Installing software conforming to 88open Consortium standards

In addition to installing software packages that conform to the DG/UX software package guidelines, you can install packages that conform to the 88open Consortium standard for software packages. Packages that conform to the 88open standard are identical in structure to AT&T System V Release 4 AIS packages.

The 88open Package menu provides operations for adding, deleting, and listing information on packages that comply with the 88open standard.

Adding 88open packages

To install packages that conform to the 88open Consortium standard for software package installation, use the **sysadm** operation `Software-> 88open Package-> Add`.

See the package's release notice for vital information such as where the package loads, how much disk space it requires, and any other procedures required to make the package usable.

This operation loads the software into the appropriate system or spool area (a directory for holding software that is loaded but not installed). If you install the package on the system, the package is immediately available for use, provided you have completed any other installation or setup procedures described in the package's release notice. Installing a package in the spool area does not allow you to use the package immediately. To make the package usable on your system, you need to invoke the Add operation again, this time specify the spool area as the device that contains the package.

Loading a package into the spool area has its advantages because it provides you with a convenient means of:

- Storing the package on-line until you are ready to complete installation on the system.
- Copying the package by allowing you to load the package from the spool area back to another portable medium such as a tape.

- Making the package available to other systems (such as OS clients) who can mount the spool directory and install the package themselves.

To remove an installed package, use the Delete operation.

Deleting 88open packages

To remove an installed 88open package, use the **sysadm** operation `Software-> 88open Package-> Delete`.

The operation removes files associated with the package and reverses any changes that the package made to other system files during installation.

The exact behavior of the Delete operation depends on scripts and file listings installed with the package that you are removing; therefore, the behavior of the Delete operation may vary from package to package. If you have created any of your own files or directories in the installed package's directories, you may want to move them (if you intend to save them) before deleting the package.

You may delete packages either from the system or from the spool area.

Listing 88open packages

To display information about installed 88open packages, use the **sysadm** operation `Software-> 88open Package-> List`.

This operation lists 88open packages on devices listed in **/etc/device.tab** or loaded on your system. By default, **/etc/device.tab** includes an entry for a tape device at **/dev/rmt/0** and an entry for the packaging spool area, **/var/spool/pkg**.

When you list packages, you may select either a detailed listing or a simple listing.

Loading and setting up other applications

The Applications Management menu provides menus and operations for loading and setting up software you install in addition to the DG/UX system and bundled software. Other software packages may or may not add menus and operations to the Applications Management menu. For more information, see the documentation for your other software packages.

Rebuilding the kernel after software setup

Adding and setting up packages may require that you rebuild and reboot the DG/UX kernel. Check the package release notice for any requirements to add or change a tunable variable in the system file. For information on building and booting a kernel, see Chapter 15.

End of Chapter

14 Setting up secondary operating systems in software release areas

In many cases, you install and run one version of one operating system on a system. If the system provides operating system (OS) services to clients, those OS clients usually run the same operating system as their OS server. When one version of the DG/UX system is the only operating system on a machine, the machine's OS services come from the directory **/srv/release/PRIMARY**, which is considered the *primary release area*. If that machine is also an OS server, its clients get their OS services from that same directory on the server.

However, you can install and run different releases of the DG/UX system on a single system. You can also install and run operating systems sold by other companies on an AViiON system running DG/UX. You install each additional release or operating system in a directory structure called a *secondary release area*, with the name **/srv/release/***release_name*.

OS clients that use a secondary release need not employ the same hardware architecture as each other or as the OS server. Though DG/UX system software is the most common type of secondary release, you can add operating systems intended for hardware other than AViiON systems.

The purpose of each release area is primarily to hold the root structures for any OS clients using the release and to hold one common copy of the **/usr** structure. The primary release area is an exception because it does not hold the OS server system's root and **/usr** file systems. Instead, it contains links pointing to the system's root and **/usr** file systems.

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 OS servers and stand-alone systems, the root is the / directory. OS clients, on the other hand, have root directories created for them under the **/srv/release/***release_name***/root** directory. OS client root directories are based on a prototype found in **/usr/root.proto**.

The number of secondary releases you can install and run on a system is restricted only by available disk drive resources and desired system performance.

The sysadm menu `Software-> Release Area` contains the operations you use to create, delete, and list release areas. This chapter explains how to perform these tasks.

IMPORTANT: The instructions in this chapter assume that you have already installed a version of the DG/UX system in the primary release area following instructions in *Installing the DG/UX™ System*.

Directories for secondary release areas

Each secondary release area directory pathname has the form **/srv/release/***release-name*, where *release-name* is the name for the operating system release stored in that area. For example, **/srv/release/dgux_54R201** might contain DG/UX 5.4 Release 2.01 and **srv/release/SunOS** might contain some version of the SunOS operating system.

Virtual disks required for a secondary release

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

To support a secondary release area for a version of the DG/UX operating system to be used by OS clients, you create the following virtual disks:

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

- If the OS client that will use the secondary release needs X11, X11 space in a secondary release area (for example, for DG/UX 5.4 Release 2.01, virtual disk **usr_opt_X11_dgux_54R201**).

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

  A difference between the **/srv/release/PRIMARY** and the **/srv/release/***secondary-release* directory structures is that the latter requires a virtual disk for the **/srv/release/***secondary-release***/usr** and **/srv/release/***secondary-release***/usr/opt/X11** file systems.

  Figure 14–1 shows an example of a primary release area (**/srv/release/PRIMARY**) and a secondary release area (all of the files within the directory **/srv/release/dgux_54R201**). In the figure, circles represent virtual disks and shaded circles represent the virtual disks mentioned above.

IMPORTANT: Although the example in this section uses DG/UX 5.4 Release 2.01 as the secondary release, remember that other DG/UX releases and other operating systems can be installed in the secondary release area.

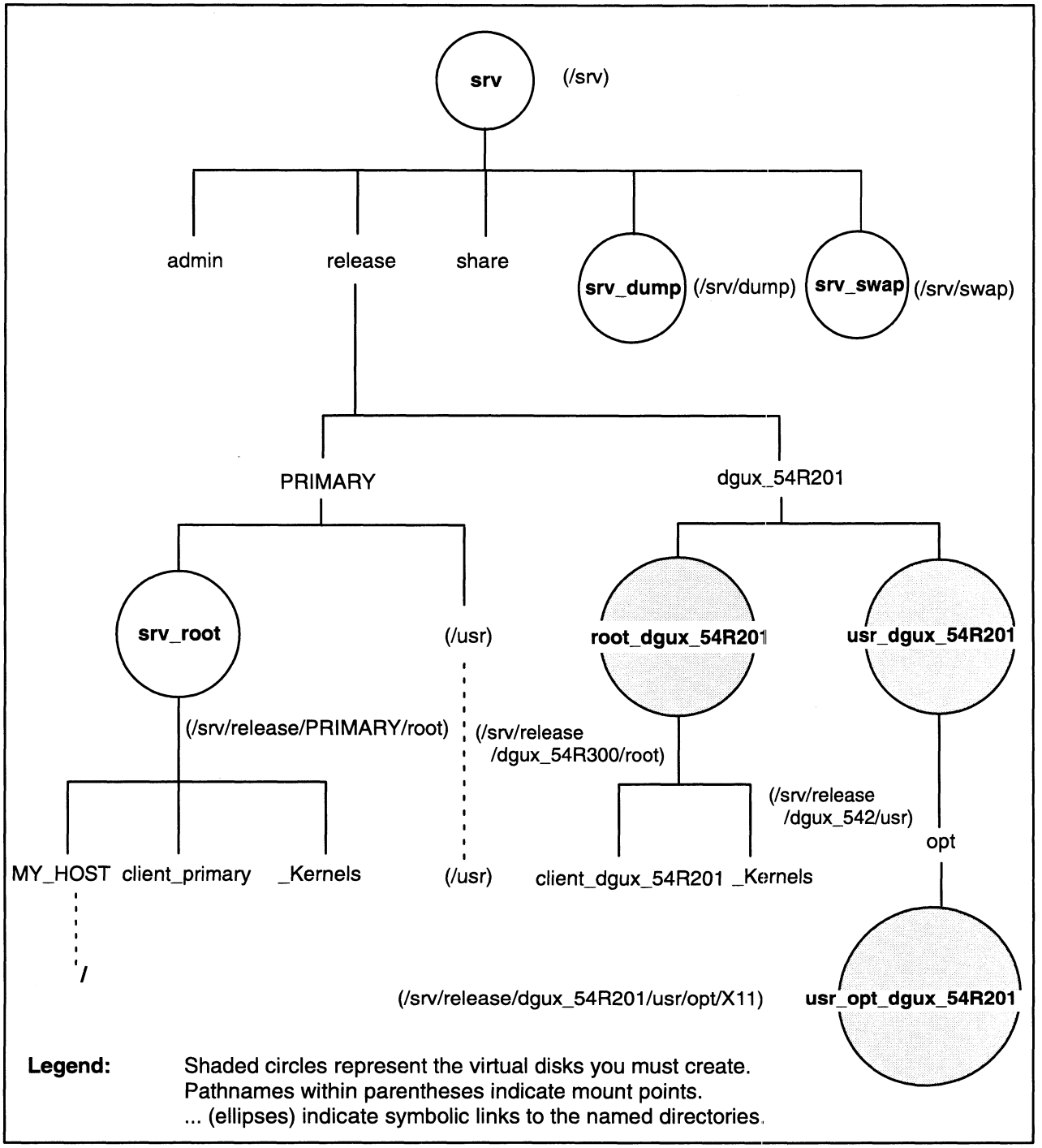


Figure 14–1 Primary and secondary release file structure

OS client root space (root_dgux_54201)

The OS client root space is a single virtual disk that contains all the root directories for all OS clients. You can mount this virtual disk wherever you want; an example of a directory mount point shown in Figure 14–1 is **/srv/release/dgux_54R300/root**. The root directory

contains subdirectories that correspond to each OS client. Instructions are provided in Chapter 18 for adding an OS client.

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

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

OS client usr space (usr_dgux_54R201)

The **usr** virtual disk, whose file system mount point shown in Figure 14–1 is **/srv/release/dgux_54R201/usr**, is reserved for system-level programs, facilities, and software packages. The **/srv/release/dgux_54R201/usr** directory holds subdirectories that contain database and configuration files, administrative commands, stand-alone utilities and bootstraps, and user commands.

The default size of the **usr** virtual disk is 240,000 blocks.

X11 package space (usr_opt_X11_dgux_54R201)

The X11 virtual disk, shown as **usr_opt_X11_dgux_54R201** virtual disk in Figure 14–1 is required only if you have purchased the DG/UX X Window Package and you intend to install the DG/UX X Window System. This virtual disk, whose file system mount point is **/srv/release/dgux_54R201/usr/opt/X11**, contains X11 documentation and an X server development environment.

The default minimum size of the X11 virtual disk is 140,000 blocks.

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

Creating a software release area

To support OS clients, first you need to create the release area that will hold the OS client software. You do not need to create a release area for OS clients that will use the primary release because the primary release area already exists, as **/srv/release/PRIMARY**. See Chapter 18 for information on adding OS clients to the primary release. This chapter covers adding a secondary release.

A release is a collection of software packages intended for a specific architecture and operating system. Adding a release means creating the appropriate directories and files that will be used by the release. Once you have added a release area, you can load software into it.

Before beginning this procedure, make sure you have enough disk space in the file system containing the **/srv/release** directory structure. For the software's space requirements, consult the release notice for the software package. Use the **df**(1M) command to display the free space in a file system. Remember that the file system reserves a 10% free space buffer. If the file system is not big enough, you need to allocate more disk space by creating one or more additional virtual disks, creating file systems on them, and mounting them in some appropriate place under **/srv/release**.

For example, if you want to add a release called **dgux5.4R3.00**, you need to make sure that **/srv/release/dgux5.4R3.00** has enough space for:

- One copy of the DG/UX system's **/usr** file system
- The OS clients' root file systems.

You could create and mount a file system at **/srv/release/dgux5.4R3.00/usr** and another at **/srv/release/dgux5.4R3.00/root**.

For how to create virtual disks and how to create, add, and mount a file system, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Use the **sysadm** operation `Software-> Release Area-> Create` to create a release area. Before you invoke the Create operation, be prepared to answer these questions:

- What will you call the release area?
- Where will you put the root directories of the OS clients of this release?
- Will you establish a directory containing software (other than **/usr**-type OS software) that all the OS clients will share? If so, where will you put this directory?
- Where will you put the OS clients' swap areas?
- Where will you put the OS clients' dump areas?

Based on your answers, the system creates the following directories and file:

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

/srv/release/*release_name***/usr/root.proto**
: The prototype for the root directories of OS clients using this release. The operation for adding an OS client makes a copy of the prototype root for each new OS client. OS 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 OS clients may hold in common.

/srv/release/*release_name***/root**
: The directory containing the root directories of OS clients. The operation for adding OS clients creates a root directory here for each new OS client, naming the directory after the OS client's host name.

/srv/swap
: The directory containing the swap areas of OS clients. OS client swap areas are named after the host name of the OS client.

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

/srv/dump/*client_name*
: The directory containing the dump files for OS clients. OS client dump files are named after the host name of the OS client.

At this point, you have created an empty release area. Using the operations under the Package menu, you now need to load the software for the release and set it up. After you have installed the software, add the OS clients. For how to add OS clients, see "Adding OS clients to the secondary release."

Deleting a release area

To delete a release area, use the **sysadm** operation `Software-> Release Area-> Delete`.

Deleting a release means deleting the release directory tree and removing files used by **sysadm** for the given release. You can only delete releases that no OS client is using. You cannot delete the PRIMARY release with this function.

Deleting a release removes the following directories and file from your system:

- **/srv/release/***release_name***/usr**

- **/srv/release/***release_name***/root**
- **/srv/admin/releases/***release_name*

Listing release information

To display the name of a release area and the pathname of the file system containing its host-independent (that is, **/usr**-equivalent) software, select the sysadm operation `Software-> Release Area-> List` operation. You can display information on all releases or on a specific release.

Installing DG/UX as a secondary release

IMPORTANT: If the server is running a previous DG/UX release as its primary release, see the documentation for that release for how to install another DG/UX version as a secondary release. Using the server's DG/UX system and **sysadm**, create a secondary release area, install the release there, and then build one or more kernels for clients who want the secondary release.

Generally, only a client can boot a DG/UX system that's in a secondary release area; the server cannot boot such a system even though it may have built the system.

Here is a summary of the tasks involved. Details on each task follow.

- Create logical or virtual disks and add file systems to hold the secondary DG/UX system release software.
- Create the secondary release area and load the secondary DG/UX release into it.
- Provide OS clients with an installer kernel and add OS clients to the secondary release.
- Write-enable the **/usr** and **/usr/opt/X11** file systems.
- Boot the installer kernel, add file systems, and set up packages.
- Customize the OS client environment by creating a custom kernel based on the secondary DG/UX release for the clients that want to use it.

IMPORTANT: For the sake of example, this section assumes that the server is running DG/UX 5.4 Release 2.01 and that the secondary release is DG/UX 5.4 Release 3.00. The instructions in this section assume the following values:

Secondary release area name	**dgux_54R300**
OS server hostname	**server_dgux_54R201**
OS client host name	**client_dgux_54R300**

Creating logical or virtual disks and adding file systems

IMPORTANT: If the server is running a release of DG/UX before 5.4 Release 3.00, you will create logical disks for the secondary release. If the server is running DG/UX 5.4 Release 3.00 or later, you will create virtual disks for the secondary release. The OS clients will use the secondary release over the network and the disk format (logical or virtual) is not significant to them.

Perform these procedures at the OS server, using the server's standard operating system.

1. Make sure you have enough disk space before you start. Table 14–1 shows the default sizes and mount point directories for the required and optional logical and virtual disks for DG/UX 5.4 Release 3.00. You need about 615,000 blocks to add just one OS client to the DG/UX 5.4 Release 3.00 secondary release. Consult the documentation for the secondary release for disk space requirements for that release.

Table 14–1 Secondary release logical and virtual disk sizes and mount points

Logical or Virtual Disk Name	Mount Point Directory	Size in 512–Byte Blocks	For One OS Client
srv_swap*	/srv/swap	(50,000 * *number–of–clients*)	50,000
srv_dump	/srv/dump	33,000	33,000
root_dgux_54R300	/srv/release/dgux_54R300/root	(*number–of–clients* * (40,000 – kernel-size)) + kernel-size)	40,000
usr_dgux_54R300	/srv/release/dgux_54R300/usr	240,000	240,000
usr_opt_X11_dgux_54R300	/srv/release/dgux_54R300/usr/opt/X11	140,000	140,000
usr_opt_networker_54R300	/srv/release/dgux_54R300/usr/ opt/networker	50,000	50,000
usr_opt_xdt_54R300	/srv/release/dgux_54R300/usr/opt/xdt	50,000	50,000
* If you already have OS clients attached to a primary release, the srv_swap disk has already been created and its file system mounted; you do not need to re-create srv_swap.			

2. Use **sysadm** to create the logical or virtual disks, with file systems on them, and sizes calculated from the table shown above.

3. Use **sysadm** to add file systems on the logical disks you just created (`File System-> Local Filesys-> Add`). Adding a file system automatically adds its entry and mount point in the file system table (**/etc/fstab**).

Creating a secondary release area

This operation creates the directory structure to hold a secondary software release. To create a release, you supply a release area name as well as pathnames identifying the locations of the **/usr** file system, the swap area shared by OS clients, root areas, and any other shared software. This operation creates the release area structure, but the release area remains empty until you load it.

1. At the OS server, follow this path through **sysadm** to create a secondary release area:

   ```
   Software-> Release Area-> Create
   ```

   Sysadm prompts:

   ```
   Release Area Name:
   ```

2. Type a unique name for the release you are adding. For example:

   ```
   Release Area Name: dgux_54R300 ↵
   Client Root Parent Directory: [/srv/release/dgux_54R300/root]
   ```

3. A default pathname for the OS client parent directory is the location for the root directories of the OS clients of this release. When you add an OS client to this release, the prototype host-dependent (/) directory structure is copied for the OS client. We suggest the default:

   ```
   Client Root Parent Directory:[/srv/release/dgux_54R300/root]↵
   /usr Directory: [/srv/release/dgux_54R300/usr]
   ```

4. Specify the pathname of the host-independent **/usr** directory for this release. Only one host-independent directory is created for each release because all OS clients of a release share the **/usr** directory. Again, we suggest the default:

   ```
   /usr Directory: [/srv/release/dgux_54R300/usr] ↵
   Share Directory: [/srv/share]
   ```

5. The **share** directory is any directory containing software the OS clients of this release share. If the directory does not exist, the operation creates it. Take the default for the **share** directory:

   ```
   Share Directory: [/srv/share] ↵
   Swap Directory: [/srv/swap]
   ```

6. Again, take the default for the **swap** directory:

   ```
   Swap Directory: [/srv/swap] ↵
   OK to perform operation?  [yes]
   ```

7. Review your answers and, if they are correct, confirm by pressing Enter:

```
OK to perform operation?  [yes]↵

Release dgux_54R300 has been added.  You may now
load software into this release area using the
Package management operations.
```

You have created a secondary release area.

Loading DG/UX into the secondary release area

Get the release media containing the secondary DG/UX release before you start these procedures. Perform the procedures at the OS server, still using the server's **sysadm**.

Follow this path through **sysadm**:

```
Software-> Package-> Load
```

Load the package into the secondary release area. You will need to supply the correct release area name; for example, **dgux_54R300**.

Providing OS clients with an installer kernel

Before OS clients can be added to the secondary release, an installer (first-time) diskless kernel must be available for booting. The kernel that performs this role is the **sysadm** kernel, copied to the default diskless kernel filename (**dgux.diskless**) and booted with the **-D** option. At the OS server, exit from **sysadm** and enter the following shell commands:

```
# cd /srv/release/dgux_54R300 ↵
# mkdir root/_Kernels ↵
# cp usr/stand/sysadm root/_Kernels/dgux.diskless ↵
```

Adding OS clients to the secondary release

1. Refer to the section "Network planning for OS clients" in Chapter 18 and complete the planning worksheets.
2. Refer to the section "Adding OS clients to the OS server's network databases" in Chapter 18 for instructions on entering the OS clients in the **/etc/hosts** and **/etc/ethers** files.
3. Refer to the section "Adding an OS client to a release" in Chapter 18 for instructions on attaching the OS clients to the secondary release. Be sure to specify the correct release area, kernel pathname, and bootstrap file. Sample values follow.

Release area: **dgux_543**
Kernel pathname: **/srv/release/dgux_543/root/_Kernels/dgux.installer.diskless**
Bootstrap file: **/srv/release/dgux_543/usr/stand/boot.aviion**

Making the /usr and /usr/opt/X11 file systems exportable

At the OS server, perform these procedures for each OS client you want attached to the secondary release. Substitute the client's name for **client_dgux_54R300** in the commands we show. The procedures performed for the first OS client you add vary slightly from the procedures performed for the subsequent OS clients.

1. Modify the **/etc/exports** file to add the secondary release area's **usr** file system and make the **usr** file system readable and writable by entering the following command to **sysadm**:

   # **admfilesystem –omodify –eP "–root=client_dgux_54R300" \↵**
   –p "rw" /srv/release/dgux_54R300/usr ↵

2. Modify the **/etc/exports** file to add the secondary release area's **usr/opt/X11** file system much as in step 1 by entering the following command to **sysadm**:

   # **admfilesystem –omodify –eP "–root=client_dgux_54R300" \↵**
   /srv/release/dgux_54R300/usr/opt/X11 ↵

3. Export the secondary release's file systems by entering the following shell command:

   # exportfs –va ↵
   `exported /srv/release/dgux_543/usr`

Booting the installer kernel, adding file systems, and setting up packages

Perform these procedures at each OS client:

1. Boot the OS client's diskless installer kernel to a run level of **i** by booting from the server. For example:

   `SCM>` **b inen( ) –D –i ↵** (Or other network controller name, such as **dgen(0)**)

2. Refer to the section "Network planning for OS clients" in Chapter 18 and complete the planning worksheets.

3. Set up the secondary DG/UX release software packages in the secondary release area. In Chapter 18, in the appropriate section for your OS client, see "Setting up packages on the OS client" for instructions on setting up packages. **Sysadm** will ask for a release area and you must specify the pathname you gave to the secondary release area instead of the PRIMARY area. In the examples, this pathname is shown as **/srv/release/dgux_54R300**.

4. Add the other file systems that are on the OS server by entering the following commands. Enter:

```
# admfilesystem -oadd -p "rw" -f \ ↵
 server_dgux_54R300/srv/release/dgux_54R300/usr/opt/networker \ ↵
  /usr/opt/networker ↵
```

5. Enter:

```
# admfilesystem -oadd -p "rw" -f \ ↵
 server_dgux_54R300/srv/release/dgux_54R300/usr/opt/xdt \ ↵
  /usr/opt/xdt ↵
```

6. Enter:

```
# admfilesystem -omount  /usr/opt/networker  /usr/opt/xdt ↵
```

7. Enter:

```
# admfilesystem -oadd -p "rw" -f \ ↵
 server_dgux_54R300/srv/release/dgux_54R300/usr/opt/x11 \ ↵
  /usr/opt/x11 ↵
```

8. Build a custom kernel for the OS client and boot it as explained in Chapter 18.

9. Log in to the DG/UX system.

   You have added an OS client to the DG/UX release in the secondary release area.

Customizing the OS client environment

After each OS client boots, continue customizing activities as desired; for example, by building one or more DG/UX kernels for them as explained in Chapter 15.

Installing an operating system other than DG/UX in a secondary release area

For how to install an operating system other than DG/UX in a secondary release area, consult the documentation and release notice for that operating system.

End of Chapter

15 Building and rebooting a DG/UX kernel

A DG/UX kernel is an executable program that provides operating system services to all other programs running on the system. The kernel runs directly on the hardware, managing access to peripherals such as tapes, terminals, and disks, as well as handling requests from users and application programs. By default, your system's current kernel is the file **/dgux**.

If you have OS clients that are not yet operational, go to Chapter 18 for instructions on building an OS client's first kernel.

When to build a kernel

Although the DG/UX system ships with a starter kernel, the file **/dgux**, you may need to build your own custom kernel to serve the specific needs of your system and users. From time to time, you may also need to rebuild your kernel to tune performance or to accommodate changes in the hardware or software configuration.

You need to build and boot a kernel for computer systems that:

- Have had any new I/O devices added
- Have had streams modules or socket protocols added
- Have had real or pseudo device drivers added
- Need a tunable variable changed

If you install new hardware or software on your system, you may have to build a new kernel. The kernel recognizes only those devices listed in the system file used to build the kernel.

If you already have a tailored kernel, but believe you need to build and boot another kernel (for example, you have just added another SCSI disk drive), check the values set in the current kernel first using the **sysdef** command as shown in "Checking the configuration variables set in the kernel." You may not need to build another kernel.

Finding out which kernel is in use

You may have multiple kernels resulting from several kernel building sessions. The default kernel is the one linked to the DG/UX system, file **/dgux**. You can find out which kernel is linked to **/dgux** by using the following shell command:

```
# ls -i /dgux* ↵
4051 /dgux   4049 /dgux.installer   4058 /dgux.gyramax
4051 /dgux.sport
```

The output shows that the inode numbers match for **/dgux** and **/dgux.sport**. (An inode is a data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID.) Therefore, **dgux.sport** is the name of the system file linked to file **/dgux**, but this is not necessarily the system running.

You can find out the kernel that is running (i.e., the kernel that was booted most recently) by running the **dg_sysctl** command as root. If you enter **dg_sysctl** with no arguments, the value of BOOTPATH is the path used for the most recent boot.

Checking the configuration variables set in the kernel

You can check the current values for configuration variables in the kernel. Checking these values may be helpful in determining whether or not you need to build a new kernel. For example, if you add a SCSI tape drive to your configuration, you must build a new kernel only if the kernel does not already include an entry for the drive.

The **sysdef** command displays values for the kernel you specify or, if you omit a kernel name, for the kernel that is linked to **/dgux**. For example:

```
# sysdef ↵
# Configured devices
#
vme()
kbd()
grfx()
syac(vme(0),0))
duart(0)
duart(1)
dgen(0)
wdt
sd(ncsc(0),0)
sd(ncsc(0),1)
sd(ncsc(0),3)
st(ncsc(0),4)
st(ncsc(0),5)
st(ncsc(0),6)

# Configuration variables
#
NODE "sport"
...
```

The list includes three SCSI tape drives on SCSI IDs 4, 5, and 6 on the first ncsc controller. If the new tape drive is connected to any of these SCSI IDs on this controller, building a new kernel is not required.

For descriptions of the configuration variables, see the file **/usr/etc/master.d/dgux**. For how to use and set these parameters to control system load and performance, see *Analyzing DG/UX™ System Performance*.

You can also peruse the system files on which kernels are based at this location: **/usr/src/uts/aviion/Build**. Scan your system files using the **more**, **cat**, **grep**, or **view** commands.

Choosing the method for building a kernel

You may select either of two **sysadm** operations to build a kernel: `System-> Kernel-> Auto Configure` **or** `System-> Kernel-> Build`. These two operations are fundamentally the same except that the Build operation lets you edit the system file before continuing with the build process. There are also a few minor differences.

In general, building a kernel involves creating a system file to reflect your hardware and software configuration. The Build or Auto Configure operation then uses the system file to determine what functionality to include in the kernel.

The system file can contain parameters that tune the operation and performance of your system. If you select the Build operation, you can add, remove, and change these parameters as you edit the system file. For more information about tunable parameters, see *Analyzing DG/UX™ System Performance*. For descriptions of these parameters, also see the **/usr/etc/master.d/dgux** file.

Besides setting tunable parameters, the primary reason for editing the system file is to verify that the devices listed there reflect the devices and device drivers installed on your system. To make this task easier, the DG/UX system offers an autosizer, **probedev**(1M), that looks for standard devices on your system and produces a list of any that it finds in standard locations. When the Auto Configure or Build operation creates a new system file, it uses **probedev** to generate the list of installed standard devices for inclusion at the beginning of the file.

If your system has nonstandard devices or device drivers, you need to add them to the list in the system file. For a complete list of devices that **probedev** recognizes, see **/usr/etc/probedevtab**.

IMPORTANT: The **probedev** program can detect only those devices that are supported by the currently running DG/UX kernel.When you create a kernel to support your current hardware, you must ensure that the underlying operating system supports all devices you could possibly have. The DG/UX installer kernel, **/dgux.installer**, and the stand-alone **sysadm**, **/usr/stand/sysadm,** support all possible devices. You must ensure that your own custom kernel supports all of your devices or that the latest revision of **/dgux.installer** is running when you create a kernel.

If you do not wish to tune your kernel and if you do not have nonstandard devices installed on your system, you will find that the Auto Configure operation is the easiest way to build a kernel. If you have nonstandard devices on your system or if you want to change the defaults of any tunable parameters, you should use Build instead.

Auto Configure builds a kernel that includes default variable values and recognizes all attached standard I/O devices. A kernel built using this option is an "auto-configured" kernel. The auto configure option requires no user interaction; it builds the kernel and links it to **/dgux** automatically.

You may choose this kernel-building option for a computer system if the system meets the following conditions:

- Does not share a disk-array storage system with another host
- Does not connect to a nonstandard I/O device
- Does not have packages (on attached disks) that require any special variable tuning

Do not use Auto Configure to build a kernel for a host that shares a disk-array storage system with another host, or is customized with nonstandard devices or tunable parameters. In this case, using Auto Configure destroys the custom changes.

Build builds a kernel that is the same as the one built with Auto Configure except that you can edit the system file, perhaps to customize the tunable variable values or add entries for nonstandard devices. A kernel built using this option is a *custom* kernel.

You should also use Build if one of the following conditions is true:

- Your system is an OS server, and you want to build a kernel for an OS client. Typically, OS servers and OS clients have different devices.
- Your system is an OS client that has local devices (such as disk or tape).

- Your system is one of the OS client hybrid configurations. An OS client hybrid configuration is one where **root** and/or **swap** reside on a local disk while the rest of the DG/UX file systems reside on another host on the network. Hybrid OS client configurations require some extra setting up, not covered in this manual. For more information on OS clients, see chapter 18.

You can build a new kernel with the Auto Configure or Build operation at any time.

If your system has exclusive access to a disk array storage subsystem (that is, your host is the only one using disks in such a subsystem), you may find the Auto Configure operation helpful for building your kernel. If two hosts use disks in the same disk array subsystem, however, you should use Build to build your kernel.

Building an auto-configured kernel

When you build an auto-configured kernel, you do not need to edit the system file. Do not use this option if any of the following is true:

- You are building a first-time kernel to support a disk-array storage system that connects to two hosts (any dual-host configuration).
- You are building a first-time kernel for an OS client at the OS server.
- You have nonstandard devices attached to your system.
- You need to tune any variables for a software package you may have installed on your system.
- You are building a kernel to add a device whose device driver is not part of the currently running kernel.
- You do not want the next kernel automatically linked to **/dgux** (**sysadm** links the kernel as part of the auto configure process).

Use the **sysadm** System–> Kernel–> Auto Configure operation to build an auto-configured kernel. The operation presents you with these queries:

`System configuration file name`

This name distinguishes the system file and the kernel from existing system files and kernels. The operation names the system file **system.***name* and places it in **/usr/src/uts/aviion/Build**, which is a symbolic link to **/var/Build**. The operation also uses this name when it creates the kernel, naming it **/dgux.***name*. You may enter a new name, or you may select the name of an existing system file.

If you select the name of an existing system file, the operation asks if you want to overwrite it by creating a new system file. If you elect to overwrite the system file, the operation also overwrites the associated kernel with the new kernel.

If you choose not to overwrite the system file, you return to the **System configuration file name** prompt.

`Operating system (OS) client`

Select this attribute if your system is a typical OS client of another host. A typical OS client has these characteristics:

- It boots the network device, **inen()**, instead of a disk.
- It mounts swap space from another host on the network.
- It mounts **root** and **usr** file systems from another host on the network.

  Do not select this attribute if your system has the operating system installed on its own disk and does not depend on another host for the services described above.

After confirming the system file name that you have specified, the operation proceeds to assemble a new system file by concatenating a list of installed standard devices (generated by **probedev**(1M)) and the prototype system files supplied with the DG/UX system and other installed packages. The operation then builds the kernel. The section "Building a kernel with Build" describes the system file and the build process in more detail.

When the build is complete, the new kernel is **/dgux.***name*, where *name* is the system file name you selected at the beginning of the operation. The kernel file **/dgux.***name* is linked to **/dgux**, which is the file that your system boots.

The new kernel will be in effect the next time you boot the system.

If you install new hardware or software on your system, you may have to build a new kernel. The kernel recognizes only those devices listed in the system file used to build the kernel.

Building a custom kernel

You use the **sysadm** System–> Kernel–> Build operation to build a custom kernel for your computer system or build a kernel for an OS client from an OS server. The Build operation is similar to the Auto Configure operation except in the following two ways:

- If the system file does not already exist, Build uses **probedev**(1M) to build a list of devices. The list is the same as the one that Auto Configure would compile if executed.
- Build lets you edit the system file before building the kernel.

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

   ```
   System-> Kernel-> Build
   System configuration file name: [xxx]
   ```

2. The system configuration filename distinguishes this system file (and derived kernel) from all other system files and kernels. The filename you type here serves as a base for all this system's files. The configuration filename will be **system**.*xxx*; the executable kernel filename will be **dgux**.*xxx*. If you want the new system file and kernel to replace existing files (which can save a lot of disk space), take the default or enter a different name that will match the names of existing files. If you want to create new files, enter a name that will produce a unique kernel filename.

   If you are creating a kernel for your machine, you will probably want to accept the default. If you are creating a kernel for an OS client, or if for any other reason you want the new system to have a unique name, then you should provide a new name. For example:

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

   Sysadm prompts:

   ```
   [system.xxx]  Correct? [yes]
   ```

3. If the new filename is the one you want, press Enter; otherwise, enter **n** and re-specify the name. For example:

   ```
   [system.xxx] Correct? [yes] ↵
   Build for this host or for OS client(s) of this
      host: [this host]
   ```

4. If you are building for this host (which may be an OS server, OS client, or neither), take the default, `this host`, and skip to step 6. If you are building an OS client kernel from an OS server, enter **OS client** and continue here. For example, if you are building for an OS client on an OS server:

   ```
   Build for this host or for OS client(s) of this
      host: [this host] OS client ↵
   ```

5. **Sysadm** prompts:

   ```
   Boot Device: [inen(0)]
   ```

   Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of an integrated network controller (**inen**), you can specify a Data General second generation network controller, named **dgen**(*n*), where *n* is a number 0 through 5. For example:

```
Boot Device: [inen(0)]   dgen(0) ↵
```

6. **Sysadm** prompts:

```
Editor: [/usr/bin/vi] ↵
```

7. Specify an editor for editing the system file. The **vi** editor is the default (see Chapter 2 for a summary of **vi** commands). Or you can can specify another editor. For **vi**:

```
Editor: [/usr/bin/vi] ↵
```

The system file consists of prototype files from the various software packages installed on your system, concatenated to form one large file. It contains entries for hardware devices, configuration variables, pseudo-devices, protocols, streams modules, tunable system parameters, and so on, as determined by the needs of the installed software packages. The file contains comments to help you understand the contents.

The system file is several screens long. You will see that configuration variables that do not apply to your system are or commented out. A line that is commented out contains a # in the first column. Please read the file, and the following sections, before you edit the file.

A common source of many kernel build failures is the absence of a comment symbol (#) used to flag notes to be ignored. Be sure you comment out all text that is to be ignored. Check your spelling and verify the DG/UX device names.

Figure 15–1 shows an excerpt from a system file.

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

Figure 15–1 Hardware devices excerpt from configuration system file

Automatically configured hardware devices

Each attached standard device (specified using the DG/UX device format with an associated brief description) is automatically configured each time you build a custom kernel. The # symbol signifies a comment, which is ignored. For information on the DG/UX device name format, refer to Chapter 20.

If you are building the kernel for this host, Build uses **probedev** to produce a list of currently configured devices. This list appears at the top of the system file. If you are building the kernel for an OS client, Build instead inserts a standard list of devices typically found on OS clients. This list appears near the beginning of the file, after some parameter settings. You should review the list of devices to verify that it reflects the configuration for which you are building the kernel.

For reference, the system file already contains entries (preceded by comment characters) for some standard devices. These entries appear in the system file under this heading:

```
##### Typical AViiON OS client device configuration
```

The complete list of standard devices in standard locations is in **/usr/etc/probedevtab**.

If you intend to install additional SCSI disks or tapes on your system in the future, you can avoid having to rebuild the kernel by

using abbreviated SCSI device specifications in the system file. Instead of specifying the SCSI ID in the device specification, use an asterisk (*). The asterisk represents all SCSI IDs on that controller. For example, instead of including these device specifications:

```
sd(insc(),0)
sd(insc(),1)
sd(insc(),2)
```

simply include this specification instead:

```
sd(insc(),*)
```

A kernel built with this specification will recognize a SCSI device at any SCSI ID on that specific controller. In general, you may include any of the following abbreviated SCSI device specifications (as appropriate for your system) in the system file:

```
sd(insc(),*)
st(insc(),*)
sd(cisc(),*)
st(cisc(),*)
sd(ncsc(),*)
st(ncsc(),*)
sd(dgsc(),*)
st(dgsc(),*)
da(hada(),*)
```

IMPORTANT: Use the asterisk this way only when you want the operating system to support all SCSI disk units it finds on the SCSI disk controller. Do not use the asterisk in any configuration where two host systems have their own set of disks in a disk-array storage system (that is, in any dual-host configuration). If you use the asterisk in such a configuration, the first host booted will take control of all the disks in the storage system and the second host will not be able to use any. For more on this issue, see the 014-series storage-system manual supplied with the disk-array storage system.

Nonstandard hardware devices

A nonstandard device uses a driver that is not supplied by Data General. You are responsible for resetting the device's jumper or DIP switch position to establish the desired device setting and for determining its device name in long form. Refer to the device's hardware installation documentation for information on resetting a device's jumper or DIP switch position, and Chapter 20 for information on DG/UX device naming conventions.

An example of a nonstandard device name in the system file follows.

```
# Hand-entered Nonstandard Devices

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

General configuration variables

NODE refers to the hostname of your computer. Your computer's hostname — for example, **sport** — is automatically supplied. It is used by the **uname** command to report the name and other attributes of the current system. Also, the **uucp** command uses the node name for performing file transfers on UNIX systems. It is also presented as part of the login banner message when you log in to your system. If you set up the TCP/IP package, the node name also corresponds to the hostname that you supply during TCP/IP setup. The node name is restricted to a maximum of 31 characters.

The master files located in the **/usr/etc/master.d** directory contain a complete list and descriptions of general configuration tunable parameters. The contents of these files determine the devices the DG/UX system can recognize and the values a number of system parameters will have. You accept the defaults by not changing any values assigned to variables in the system file. The information for the DG/UX system itself is in the file **dgux**. If you have additional products, such as TCP/IP or ONC/NFS, their configuration files are in this directory also.

IMPORTANT: You may view the master files in **/usr/etc/master.d**, but do not edit them. Instead, override a given default by setting the parameter to the desired value in the system file. Do not duplicate the master files in the master directory, or you will not be able to build your kernel.

A sample tuning variable is:

```
MAXUP 60
```

MAXUP specifies the maximum number of processes a non-superuser can have at any one time. The default MAXUP value is 50 processes. The preceding example overrides the default value.

For more information on tuning parameters, including a discussion of the parameters that you are most likely to want to change, see *Analyzing DG/UX System Performance*.

Package variable tuning

The system file contains a series of concatenated package prototype system file fragments. If you have just loaded a new package on

your system, check its release notice for information on possible variable tuning. If you need to tune a variable, either enter or modify it in the system file. Where you place the variable and value pair in the system file is unimportant. For example:

```
# These variables set values for message parameters and inter-
# process communication shared memory for fredware package.

MSGMAX      8192
MSGMNB      16384
SHMMAX      524288

# End of fredware variables.
```

These variables affect space for messages and shared memory for interprocess communication. MSGMAX specifies the number of bytes a message can contain; MSGMNB specifies the maximum number of bytes a message queue can contain; and SHMMAX specifies the maximum number of bytes in a segment. The package's release notice specifies certain variable value assignments. The example above shows kernel variables tuned for specific needs.

OS client configuration variables

If you are building a kernel for an OS client at the OS client, you must insert these OS client configuration variables and assigned values. You can do so anywhere in the system file. For an OS server building a kernel for an OS client, these variables are set automatically.

```
# OS Client Configuration Variables

NETBOOTDEV  "inen( )"
ROOTFSTYPE  NETWORK_ROOT
SWAPDEVTYPE NETWORK_SWAP
```

NETBOOTDEV, ROOTFSTYPE, and SWAPDEVTYPE provide resources to the OS client over the network. Make sure there are no leading comment symbols (#) in the first columns of the lines containing these variables. Also, comment out the descriptive line labeled "OS Client Configuration Variables" as shown above.

Tunable configuration variables for workstations

Kernels built for systems with a graphics device and those built for OS clients may need special tuning to improve system performance. The MAXSLICE variable is set automatically on these systems.

```
MAXSLICE 50
```

MAXSLICE specifies the maximum time in milliseconds a user process can run before being suspended. After a process executes for its allocated time slice, that process is suspended. The default MAXSLICE value is 500 milliseconds (1/2 second). Reducing the MAXSLICE value can improve interactive response time for users running the X Window system. The preceding example overrides the default value.

8. After you finish editing the system file, save the file and exit from the editor (**ZZ** command for **vi**).

   After you edit the system file, the next step depends on whether you are building the kernel for an OS client of this host or for this host. If you are building the kernel for **OS client(s)**, the operation now tries to build the kernel. If you are building the kernel for **this host**, the operation presents the following prompt:

   ```
   Link the new kernel to /dgux
   ```

   You must let **sysadm** link the new kernel (here, named **dgux.sport**, for example) to **/dgux** before the new kernel can be booted with the default boot path. If you do not link the new kernel to **/dgux**, the existing kernel (if one exists) remains linked to **/dgux**, and you continue to use the old kernel.

   After building the kernel executable, the operation moves it to the root directory as **/dgux.***name*. If you choose to link the new kernel to **/dgux**, the operation creates a link (using the **ln**(1) command) from **/dgux** to the new kernel so that the new kernel will be the one that boots when you next boot the system.

9. To link the new kernel to **/dgux**, accept the `yes` default. If you want to keep the current kernel linked to **/dgux**, enter **n**. For example:

   ```
   Link the new kernel to /dgux? [yes] ↵
   ```

   If you specified the name of an existing system earlier, **sysadm** asks if you want to save the existing system files. If you specified a new name, **sysadm** skips the following prompt:

   ```
   Save the old kernel? [yes]
   ```

10. If you want to save the existing system files in addition to the new ones (at some cost in disk space), answer yes; to delete the existing system files and replace them with the new ones, enter **n**. For example:

```
Save the old kernel? [yes] n ↵
```

Sysadm prompts:

```
Continue with the build?  [yes]
```

11. To continue, confirm:

```
Continue with the build?  [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.xxx
```

If you told it to link **/dgux**, **sysadm** also displays:

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

To build the kernel, the operation first runs **config**(1M) on the system file and produces program code in a file named **conf.c**. After **config** finishes, the operation compiles **conf.c** and links the libraries in **/usr/src/uts/aviion/lb** to build the new kernel image. After successful completion, the bootable kernel file is in either of two places:

/ If you built the kernel for **this host**, the kernel is in the root directory, as **dgux.***name*. If so directed, the operation also linked the kernel to **/dgux**.

/usr/src/uts/aviion/Build
If you built the kernel for **OS client(s)**, the kernel is in this directory, which is a link to **/var/Build**. The kernel is named **dgux.***name*. You should move the kernel to some directory that is accessible to the OS clients.

Typically, OS client kernels reside in **/srv/release/PRIMARY/root/_Kernels**, or the equivalent directory for a secondary release. Assuming that the OS client root directories are in the same file system as the **_Kernels** directory, you can now create links (using **ln**(1)) from **dgux** in the OS client root directories to the new kernel in **_Kernels**. For example, after moving OS client kernel **dgux.diskless** to the **_Kernels** directory, type the following to make it available to OS client **goober**:

```
# cd /srv/release/PRIMARY/root/goober ↵
# ln ../_Kernels/dgux.diskless dgux ↵
```

IMPORTANT: If the build fails, see "Configuration error messages," correct the problem, and invoke Build again. Remember that many build

failures result from the absence of a comment symbol (#) in comment lines in the configuration file.

You have successfully built a custom kernel. The kernel file resides in **/dgux.***xxx* (*xxx* is the name you specified in step 2). The system configuration file resides in **/usr/src/uts/aviion/Build/system.***xxx*.

You must reboot the system for the new kernel to take effect.

For more information on setting up OS clients, see Chapter 18.

Configuration error messages

The following error messages are generated by the **config**(1M) program during the kernel building process. 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 do not alter the master file. You alter the master file only if you install a new device driver.

`A master file entry for` *`entry`* `already exists.`

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.

`Cannot open the master file [`*`master_file_name`*`].`
`Cannot open master file directory.`

Cannot open a file or directory. Make sure the master file is in the proper directory and that it is named correctly.

`No section definition found in master file [`*`master_file_name`*`].`
`This file will be ignored.`

The file in the master directory is not a legal master file.

`Unknown Keyword: [`*`keyword_name`*`]`
`Unknown Device Flag: [`*`device_flag`*`]`
`Unknown Flag: [`*`flag`*`]`

There may be incorrect information in your system file, such as a misspelled 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.

```
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.
```

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.

```
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.
```

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.

Rebooting a kernel

IMPORTANT: Regardless of whether you build the kernel at the OS server or OS client, each OS client must boot its own kernel.

When you reboot a kernel, all processes running on the system are killed. Therefore you should not do this if users are logged on and/or applications are running.

Select the **sysadm** operation `System-> Kernel-> Reboot` to reboot the kernel. The operation shuts down the system completely (except for the hardware) and restarts the operating system, activating the new kernel.

1. From the **sysadm** System–> Kernel menu, select the Reboot option. **Sysadm** displays the boot path, for example:

   ```
   Boot path: [sd(ncsc(0),0,0)root:/dgux -3]
   ```

2. Verify the boot path, entering the name of the new system if it differs from the one displayed. Details on specifying a boot path appear in Chapter 3. Press Enter. **Sysadm** asks for confirmation:

   ```
   All currently running processes will be killed.
   Are you sure you want to reboot the system? [yes]
   ```

3. Confirm your desire to reboot by pressing Enter again.

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

```
Booting sd(ncsc(0),0,0) root:/dgux -3
DG/UX Bootstrap 5.4R3.00
Loading image ..........................

sport
Console login:
```

   In the following example, the kernel **/dgux** on the **root** virtual disk located on **sd(insc(0),0,0)** is booted automatically to run level 3. You could specify a different run level.

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

   When you finish rebooting the kernel, you can boot and use the system. For how to boot the system, see Chapter 3.

Changing configuration variable values to handle your system load

System configuration variables in the system file set various table sizes and system thresholds to handle the expected load on your system. When you edit the system file as part of the rebuilding the kernel, you can change the values of these parameters.

The default system parameter values are adequate for most configurations and applications. If your applications have special performance needs, you can experiment with different combinations of values to find an optimal set to support your computing environment.

For descriptions of the system parameters, see the file **/usr/etc/master.d/dgux**. For how to use and set these parameters to control system load and performance, see *Analyzing DG/UX™ System Performance*.

End of Chapter

16 Accounting

Through **sysadm**, you can print or display on your terminal system-use data that is logged and then stored in summary files and reports. These reports are useful for keeping track of system use by login, command, or machine.

This chapter shows you how the components of the accounting system function.

Starting and stopping accounting

By default, accounting is turned off. To start the accounting system, execute the **sysadm** operation `System-> Accounting-> Start`. To turn accounting off, execute `System-> Accounting-> Stop`.

Turning on accounting starts accounting immediately. It also arranges to have accounting run when the system reboots and arranges for the following accounting commands to be run as regular jobs with **cron**:

```
0 4 * * * /bin/su - adm -c "/usr/lib/acct/runacct 2> \
   /var/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
```

The default jobs are described in Table 16–1.

Table 16–1 Regular accounting jobs

Job	Description
runacct	Runs at 04:00 a.m. to collate statistics on connects, user activity, CPU usage, fees, disk usage, and so on. Error messages go to **/var/adm/acct/nite/fd2log**.
ckpacct	Runs at 5 minutes after every hour to monitor the size of the **pacct** data repository file, renaming it when it reaches its size limit (by default, 500 blocks).

Continued

Table 16–1 Regular accounting jobs

Job	Description
monacct	Runs at 05:15 (5:15 in the morning) on the first day of every month to collate and summarize monthly statistics. The **monacct** program 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.
dodisk	Runs at 02:00 a.m. to perform disk accounting.

To change the default values for accounting **cron** jobs, execute `System-> Accounting-> Cron-> Change`, which invokes an editing session with an editor such as **vi**, allowing you to hand-edit the **cron** accounting prototype file.

Listing the accounting reports

You can obtain accounting reports with the **sysadm** operation `System-> Accounting-> List`, from which you select one of these report types:

- Command usage
- User logins
- Full report

The full report has the following sections:

- Daily line usage
- Daily usage by login name
- Daily total command summary
- Monthly total command summary
- Last login

Each time you request a full listing, you generate all of these reports automatically. You cannot generate a single report.

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.

The following is an example of a daily line usage report:

```
Nov 11 16:00 1993  DAILY REPORT For DG/UX page 1

from Tue Nov 10 08:27:00 1993
to   Wed Nov 11 04:00:42 1993

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
```

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.

Continued

`# 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 a port service is first enabled on a port 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 `# 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.

Daily usage by login name

The following is an example report for each user:

```
Nov 11 04:42 1993 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
```

The daily usage by login name report gives a breakdown of system resource use for each user:

`Uid`	User ID.
`Login Name`	Login name of the user. There can be more than one login name for a single user ID.
`CPU (Min)`	CPU time used, divided into `Prime` and `NPrime` (nonprime). See "Updating holidays" for information about prime and nonprime time.
`Kcore-Mins`	Total memory a process used, in kilobytes per minute. Divided into `Prime` and `NPrime`.
`Connect(Min)`	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`	Total amount of disk I/O performed by a user login.
`# 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. You use the **chargefee**(1M) command to charge a user for special services, such as restoring files and mounting tapes.

Daily and monthly total command summaries

The daily and monthly total command summaries 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 summary reflects the data accumulated since the last invocation of **monacct**.

These summaries tell which commands are used most. Since you know which system resources the commands use, you can tune the system accordingly. The summaries are sorted by `Total KCoremin` (see below), which is a good way to calculate drain on a system.

The following are examples of the two types of command summaries. First is an example of a daily command summary:

```
Nov 11 04:42 1993 Daily Command Summary

Command   Numb  Total    Total   Total     Mean    Mean     Hog     Chars   Blocks
Name      Cmds  KCoremin CPU-Min Real-Min  Size-K  CPU-Min  Factor  Trnsfd  Read

TOTALS    2332  1624.74  16.05   15210.91  5.67    003      0.01    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
```

Next is a monthly command summary:

```
Nov 11 04:42 1993 Monthly Total Summary

Command   Numb   Total      Total   Total      Mean     Mean    Hog     Chars   Blocks
Name      Cmds   KCoremin   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

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
```

The columns in the report are:

`Command Name`	The name of the command. All shell procedures are reported under the specific shell's name such as **sh**, **csh**, and **ksh**.
`Numb Cmds`	Number of times a command was invoked.
`Total KCoremin`	Total kilobyte 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 CPU time (`Total CPU-Min/Numb Cmds`).
`Hog Factor`	Ratio of system availability to system usage (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.
`Blocks Read`	Total physical block reads and writes that a process performed.

Last login

This report gives the date when a particular login name was last used. The report can help you find likely candidates for the tape archives, such as **/usr** directories associated with unused login names.

The following is an example report:

```
Nov 11 04:42 1993 LAST LOGIN

00-00-00        bin     92-11-02        carson

00-00-00        croot2  92-10-09        moe

00-00-00        daemon  92-11-11        larry

00-00-00        svvs    92-11-10        curly

00-00-00        archive 92-11-01        tlp

91-11-12        poulet  92-11-11        uucp
```

A field of all zeros means that login was not used since the last invocation of the **lastlogin** program.

Updating holidays

You can update your holiday database with the **sysadm** operation `System-> Accounting-> Edit Holidays File`, which invokes an editing session with an editor, such as **vi**, on the file **/usr/lib/acct/holidays**.

The table format has three types of entries:

- Comment lines

  Comment lines have an asterisk in column 1.

- Year designation line

  This line should be the first data line (a line that is not commented) in the file; it must appear only once. It has three fields: year, prime time, and nonprime time. For example, to specify the year 1992, prime time at 8:30 a.m., and nonprime time at 5:00 p.m., enter:

  ```
  1992    0830    1700
  ```

  The time 2400 is automatically converted to 0000.

- 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 following is an example of a **/usr/lib/acct/holidays** file:

```
*
* Copyright (C) Data General Corporation, 1984 - 1992
* 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: <@(#) holidays,v 4.1.1.2> $
*
*
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr     Prime     Non-Prime
* Year     Start     Start
*
  1992     0830      1700
*
* Day of   Calendar  Company
* Year     Date      Holiday
*
    1      Jan 1     New Year's Day
  146      May 25    Memorial Day
  185      Jul 3     Independence Day
  251      Sep 7     Labor Day
  331      Nov 26    Thanksgiving
  332      Nov 27    Day After Thanksgiving
  360      Dec 25    Christmas Day
  366      Dec 31    New Years Eve
```

Accounting commands

You can invoke the accounting commands described in Table 16–2 from the shell.

Table 16–2 Accounting commands for use from the shell

Command	Function
acctcon1	Reads **/etc/wtmp** and converts login and 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), nonprime connect time (seconds), session starting time, and starting date and time.
acctdusg	Computes disk resource consumption (including indirect blocks) for each login. See **acct**(1M).

Table 16–2 Accounting commands for use from the shell

Command	Function
acctwtmp	Records boot times in **/etc/wtmp**.
chargefee	Charges a specified amount against a specified user by generating a charge report and logging it to **/var/adm/fee**. The **runacct** program reads this charge and merges it into the total accounting records.
ckpacct	Checks and controls the size of **/var/adm/pacct**. When **pacct** grows larger than 500 blocks, **turnacct "off"** turns **accton** off, renames the current **pacct** file to **pacct1**, and creates a new **pacct**. Finally, **ckpacct** turns **accton** back on.
dodisk	Does disk accounting on the special files in **/etc/fstab**. Creates **/var/adm/dtmp.**
monacct	Uses the daily data organized by **runacct** and writes it into a monthly summary. Invoke **monacct** through **cron** once each month or each accounting period. **Monacct** creates summary files in **/var/adm/acct/fiscal**. See **acctsh**(1M).
runacct	The main shell program for daily accounting. See **runacct**(1M) for details.
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.

Recovering from runacct failure

If the system crashes, **/var** 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. These files are located in **/var/adm/acct/nite**.

Runacct may produce the following error messages. We suggest ways to recover from them.

`acctg alreadyn 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 according to instructions in "Fixing corrupted files."

`locks found, run aborted`

The files **lock** and **lock1** were found in **/var/adm/acct/nite**. Remove these files before restarting **runacct**.

`Spacct`*`string`*`.`*`MMDD`* `already exists`

File setups have already run. Check status of files, then run setups manually according to instructions in "Restarting runacct."

`turnacct switch returned rc=`*`string`*

Check the integrity of **/usr/lib/acct/turnacct** and **/usr/lib/acct/accton** by ensuring that the **accton** program is owned by **root** and has the **setuid** bit set. See **chmod**(1).

`/var/adm/acct/nite/wtmp.`*`MMDD`* `already exists, run setup manually.`

Run setups manually according to instructions in "Fixing corrupted files."

`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 to specify the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of **/var/adm/acct/nite/statefile**. To override **statefile**, include the desired state on the command line. As mentioned 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**, enter:

```
# nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart **runacct** specifying *MMDD* (0601), enter:

```
# nohup runacct 0601 2> /var/adm/acct/nite/fd2log &
```

To restart **runacct** at a specific state, such as **wtmpfix**, enter:

```
# 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 **/var/adm/acct/nite/fd2log**; check this file periodically for error messages. Specifying *MMDD* creates a new **/var/adm/acct/nite/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 **/etc/tacct** files yourself.

Fixing wtmp errors

During normal operation, monitor the size of **/etc/wtmp**, which is the basis for the connect accounting. 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.

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 the file:

1. **Go to directory /var/adm/acct/nite.**
2. **Enter the following command:**

```
# fwtmp < wtmpMMDD > xwtmp ↵
```

This command line executes the **fwtmp**(1M) program on the contents of **wtmp***MMDD* and stores the output in file **xwtmp**, effectively converting the binary contents of **wtmp***MMDD* into ASCII format.

3. **Use an ASCII editor such as vi to delete all remaining binary lines from xwtmp.**

4. **When you're finished editing, convert from ASCII back to binary with the following command:**

```
# fwtmp -ic < xwtmp > wtmp.MMDD ↵
```

If the **wtmp** file is beyond repair, create a null **wtmp** file by issuing a command like this as superuser:

```
# > /etc/wtmp ↵
```

Cleaning out **wtmp** prevents any charging of connect time. In general, you should check the size of **wtmp** occasionally to see if it is taking up too much space. Reduce the size of **wtmp** either by emptying it as shown above, or by truncating it. To truncate **wtmp**, leaving only the last 3200 characters (the last 50 entries), issue these command lines:

```
# tail -3200c /etc/wtmp > /tmp/wtmp ↵
# mv /tmp/wtmp /etc/wtmp ↵
```

Fixing tacct errors

If you are using the accounting system to charge users for system resources, the integrity of **/var/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 **/var/adm/acct/sum/tacctprev** with **prtacct**. If **prtacct** does not report any errors, patch up **/var/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:**

```
# acctmerg -v < tacctMMDD > xtacct ↵
```

This command line executes the **acctmerg**(1M) program on the contents of **tacct***MMDD*, and stores the output in file **xtacct**, effectively converting the binary contents of **tacct***MMDD* into ASCII format.

3. **Edit xtacct and delete all corrupted records.**

4. **When you've finished editing, convert from ASCII back to binary with the following command:**

   ```
   # acctmerg -i < xtacct > tacctMMDD ↵
   ```

   Remember that **monacct** removes all the **/var/adm/acct/sum/tacct***MMDD* files. Therefore, when you merge these files together, you recreate **/var/adm/acct/sum/tacct**.

End of Chapter

17 Security

The DG/UX system supports a number of security features. For environments where you require a greater degree of security, however, there are also the Trusted DG/UX systems, which provide B1 and C2 levels of security. For information on the Trusted DG/UX systems, contact your Data General representative.

General recommendations

In general, you may find the following suggestions helpful 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. Password aging, discussed in Chapter 5, is a feature that can force users to change passwords regularly. Advise users not to pick obvious passwords. Users should avoid passwords that common "password-cracker" programs may guess, such as proper names and any word appearing in the dictionary. In addition, the **passwd**(1) command, used to set passwords, imposes other restrictions.
- All port services, whether served through a terminal or a modem, should run **login** or another service that requires password validation before granting access to the system.
- 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, personal configuration files, such as **.profile**, **.login**, and **.cshrc**, and files in **/usr/sbin**, **/usr/bin**, and **/etc** should not be writable by others.
- Encrypt sensitive data files. The **crypt**(1) command and the encryption capabilities of the editors (**ed** and **vi**) provide protection for sensitive information. Encryption/decryption capabilities are available as a separate product with only U.S. releases of the DG/UX system. Contact your Data General representative for more information.

- Do not leave a logged-in terminal unattended, especially if you are logged in as **sysadm** or **root**.
- To check your file systems for files that may indicate security breaches, use the **sysadm** operation `File System-> File Information-> Check`. The operation looks for two kinds of files:

  Device files outside of /dev
  Device files, normally located in the **/dev** directory, provide access to peripheral devices on your system. The Check operation looks for device files in directories other than **/dev** because such files may provide unauthorized access to the data on a device.

  Setuid executables owned by the superuser
  Executables (programs and scripts) that have the setuid bit set will run under the username of the program's owner rather than with the username of the invoking user. If such an executable is owned by the superuser (the **root** or **sysadm** profile), the program will run as a superuser process, regardless of who invokes it. Depending on the nature of the program, it may give an unauthorized user access to sensitive data or applications.

  For information on the Check operation, see "Checking file systems for security breaches."

- Do not put the current directory, represented by a dot (.), on any superuser search path. In user search paths, the current directory should always appear last, if at all. Placing the current directory on your path may cause you to execute inadvertently an nonsecure script or program that has the same name as a common command.

Default shell and restricted shell

Generally, when a user logs in the default program that is started is **/sbin/sh**. There may be cases, however, where a user needs to be given a restricted shell, **/bin/restsh**. A restricted shell is one where the user is not allowed to:

- Change directories.
- Change the value of PATH.
- Specify pathnames 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.

Restricting access to user-created files

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**). The default for files gives all users read and write permission. For directories, all users get, write, and execute permission. Execute permission on a directory lets you make the directory your working directory and list its contents.

If you consider the default permissions too permissive, you can set your own defaults by including the **umask** command, with some appropriate argument, in your personal profile. The **umask** command alters your default permission levels by the amount specified in the argument. The argument is a three-digit number 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.

The following line, for example, would reduce the default owner permissions by 0, the default group permissions by 2, and the default permissions for others by 7:

```
umask 027
```

Thus the resulting default for directory creation is to set permissions to 750, and for file creation, 640. See the **chmod** manual page for more information on permissions.

Changing a password

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

- Your password must be different from your login name.
- Your password should not be any obvious rearrangement of the characters in your login name. For example, if your login name is **anemone**, do not use the password **nemonea**.

- The new password must have at least six characters. At least two characters should be uppercase or lowercase alphabetic characters (a-z and A-Z), and at least one character should also be a numeric or special character, such as 0 through 9, _, –, ?, !, @, $, or space.

Password aging

To enforce a basic level of security on a system, you can require that users have passwords to access their login accounts. A password aging mechanism is also available to force users to change passwords periodically and prevent them from changing a password before a specified time interval. If password aging is not used, a person can keep the same password indefinitely.

For how to activate the password aging, see Chapter 5.

Checking file systems for security breaches

Select the **sysadm** operation `File System-> File Information-> Check` to search a directory tree for files that have suspicious ownership and permission settings. If you specify no directories, the operation checks the system's entire directory tree. This operation may be time consuming.

The Check operation finds files that may indicate that 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 created or moved device files for a special purpose, such as a test.
- Non-system files that are owned by the superuser (user with UID of 0, **sysadm** and **root** by default) and have the setuid bit set.

The setuid bit is a special kind of permission attribute that all files have but which is only useful for executable files (such as programs and shell scripts—not directories or text files). When a user runs an executable that has the setuid bit set, the process runs with the effective user ID of the executable's owner — not, as is normally the case, with the user ID of the person running the executable.

For example, if user **fred** executes a program owned by user **bob**, and this program does not have the setuid bit set, **fred**'s process runs with the effective user ID of **fred** (as normal). On the other hand, if user **fred** executes a program owned by user **bob**, and this program does have the setuid bit set, **fred**'s process runs with the

effective user ID of **bob**. This means that the program, if it is so written, can access files to which **fred** may not normally have access, but to which **bob** does. User **fred** does not even need to know **bob**'s password for these accesses to occur.

The rationale behind the setuid bit is to allow users to perform some action that they should not be able to perform under normal circumstances. The **lp** command, for example, has the setuid bit set so that any user who invokes **lp** to queue up a print job can, through the agency of the **lp** program, do things such as copy files to the LP system's directories and add requests to the LP scheduler's queue file.

When you execute a program that has the setuid bit set, your actions are determined by the scope of the program and the user ID of the program's owner. Thus, 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 Check operation includes a search for suspicious programs, ones owned by **root** that have the setuid bit set.

The following example file listing shows **ls –l** output for several files that have the setuid bit set (which we know because of the **s** flag in the group-execute permission and/or owner-execute permission places in the permissions line). The file **/sales/tom/nasty** is suspicious because it is owned by **root** but is obviously not a normal system program. It appears to belong to user **tom**. Depending on what Tom's program does, it may constitute a security breach.

```
-rwsr-sr-x  1 root  bin    44924 Mar 28 18:16 /usr/bin/at
-rwsr-sr-x  1 root  bin    29500 Mar 28 18:16 /usr/bin/crontab
---s--x---  1 root  users  95376 Aug 18 11:08 /sales/tom/nasty
```

Tom would never have been able to create such a program without superuser access. Thus, the program may not only constitute a security breach itself, but it also indicates the presence of a breach elsewhere, the one that allowed Tom to become superuser and produce the suspicious executable.

To protect your system, never leave your logged-in terminal unattended (particularly if you are logged in as **root** or **sysadm**). Another user could move or copy files, or commit any manner of destructive acts, all with your user ID.

When you find a setuid bit set, investigate further. You may need to correct setuid permissions with the **chmod** command. In general, you will not be creating device files, so none should exist outside 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.

For environments where you require a greater degree of security, there are the Trusted DG/UX systems, which provide B1 and C2 levels of security. For more information on the Trusted DG/UX systems, contact your Data General representative.

Protecting files in shared directories

You can protect files in directories accessible to all users from deletion by users who do not own them by setting the sticky bit for that directory. Without the sticky bit set, other users who do not have ownership or access to the file can remove it if they have write access to the containing directory.

For how to set the sticky bit for a directory, see Chapter 4.

End of Chapter

18 OS servers and clients

This chapter tells how to make OS (operating system) client computers operational, and how to manage OS clients. An OS client is a workstation that depends on another system, called the OS server, for its operating system software. The OS server supplies a bootable operating system and file system space over a local area network (LAN) to an OS client.

The OS client boots its network controller, thereby initiating a series of network transactions with its OS server. The OS server transfers a kernel image to the OS client so the client can boot and mount remote file systems.

The **sysadm** Client–> OS Client menu provides menus containing operations for managing OS clients. The OS Client operations apply only if you set up an OS server and clients.

IMPORTANT: This chapter assumes that the OS clients will run DG/UX 5.4 Release 3.10 from the server's primary release area and that the OS server is running a DG/UX 5.4 Release 3.10 kernel. If you want to connect an OS client to a secondary release area, see "Running a secondary operating system on an OS client."

Managing OS clients

The **sysadm** Client–> OS Client menu provides operations for adding a client to a release, deleting a client from a release, modifying a client's bootstrap link, listing clients attached to a release, and setting a client's current release. The OS Client menu also provides the Defaults menu for managing the values that appear as the defaults when you add a client.

Types of OS clients

The OS server supplies a bootable operating system and file system space over a local area network (LAN) to an OS client. The OS client comprises the **root** and **usr** file systems and **swap** area which are typically exported from the OS server as the **/srv/release/PRIMARY/root**, **/usr**, and **/srv/swap** file systems. In addition, **/srv/dump** is needed for OS clients. The **srv_root**, **usr**, **srv_swap**, and **srv_dump** virtual disks are needed for these file systems and special areas. Normally, an OS client with no attached disk drive receives its entire operating system and file system space

from an OS server. However, if an OS client has an attached disk drive, you may choose to put part of the operating system on the local disk drive to maximize system resources and to improve system performance. The three required virtual disks (**root**, **usr**, and **swap**) can be divided between two disk drives, one attached to the OS server and the other to an OS client. The three supported types of OS clients are

Diskless The OS client has no disk drive and receives its entire operating system and file space from the OS server over a LAN.

Local root and swap An OS client with an attached disk drive provides its own local **root** and **swap** virtual disk resources, while receiving its **usr** virtual disk resources from the OS server via a LAN.

Local swap An OS client with an attached disk drive provides only its own local **swap** virtual disk resources, while receiving its **root** and **usr** virtual disk resources from the OS server via an LAN.

Calculating OS server virtual disk requirements for diskless OS clients

Before you add diskless OS clients, you must create the following virtual disks on the OS server to accommodate them:

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

This section describes each required virtual disks and how to calculate sizes for the disks. For an explanation of virtual disks and how to add the virtual disks, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Figure 18–1 is a worksheet you can complete as you decide which devices will contain the virtual disks and how large to make each virtual disk. Figure 18–2 is a completed example of the worksheet.

OS server virtual disk requirements

Virtual Disk or Mirror Name	Mount Point Directory	Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes	
		Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks
srv							
srv_root							
srv_dump							
srv_swap							
Total Used							
Total Capacity							
Free Space							

Figure 18–1 OS server virtual disk worksheet

OS server virtual disk requirements

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

Figure 18–2 OS server virtual disk worksheet, completed example

Layout of the required virtual disks

Figure 18–3 shows **/srv/release/PRIMARY**, the DG/UX primary release area. In the figure, shaded circles represent the virtual disks you must create on the OS server for its OS clients.

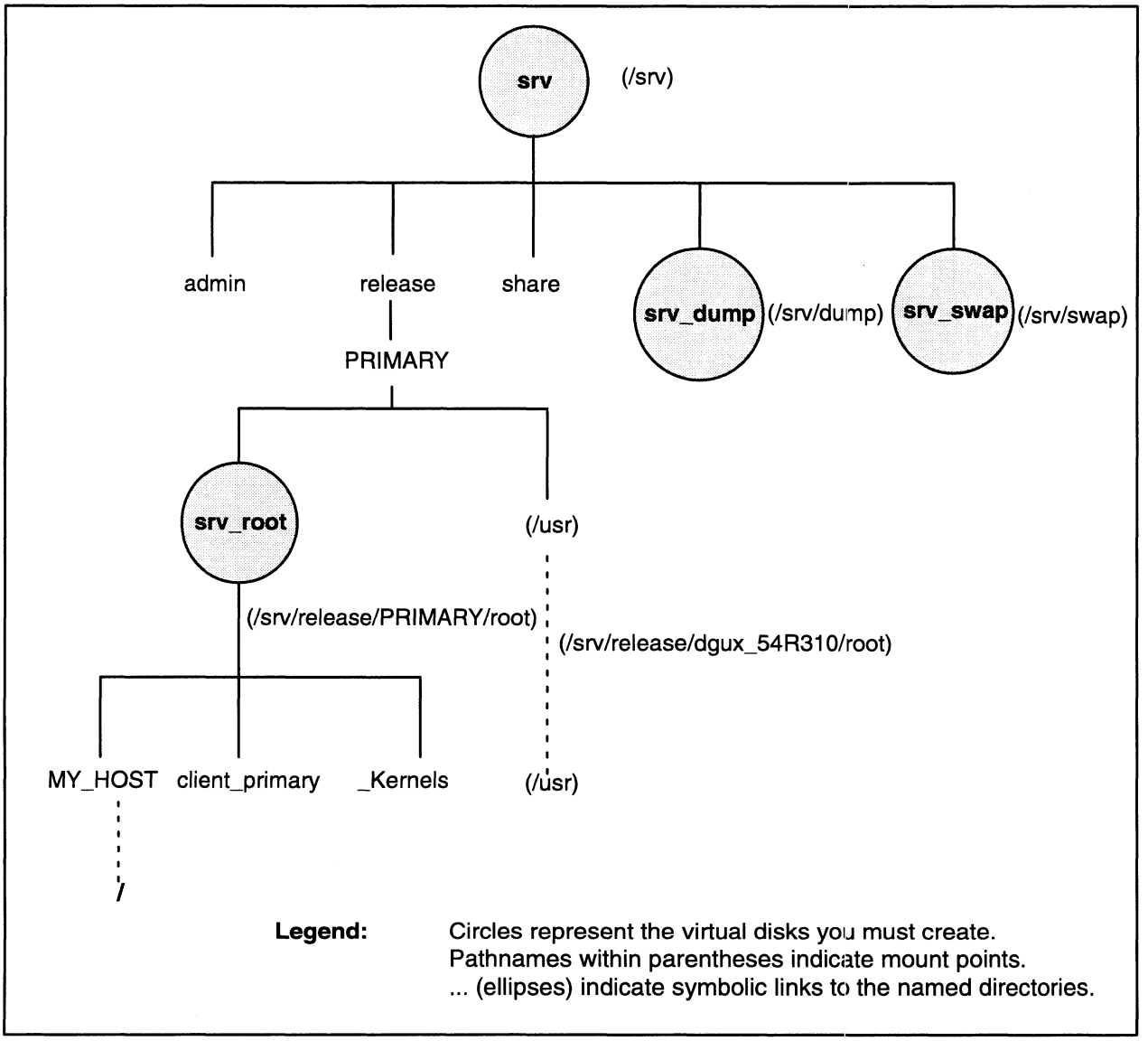


Figure 18–3 /srv file system

OS client directory (srv)

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

The **srv** virtual disk should be large enough to accommodate the **sysadm** database. You should create a single virtual disk named

srv with a size of 5,000 blocks. You do not need to add additional space for this virtual disk.

OS client root space (srv_root)

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

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

Root space for OS clients with individual kernels

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

Root space for OS clients sharing a common kernel

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

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

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

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

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

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

Do not add any additional space as overhead.

OS client swap space (srv_swap)

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

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

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

Use the following formula and example to compute the initial swap space for OS clients:

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

The number of *blocks-per-OS-client* is either 50,000 (24 Mbytes) or 1.5 times physical memory, whichever is larger. For example, for four clients, the numbers would be:

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

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

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

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

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

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

OS client dump space (srv_dump)

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

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

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

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

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

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

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

Running a secondary operating system on an OS client

Read this section only if you want the OS clients in your configuration to run a DG/UX release other than the primary one installed on the server, or if you want the OS clients to run an operating system other than DG/UX.

In addition to providing a version of the DG/UX operating system as the primary release to its OS clients, an OS server can provide other (secondary) releases of DG/UX or another UNIX system.

The DG/UX file system can support separate operating systems in different release directories on a single server. The DG/UX release you load with the standard installation instructions in *Installing the DG/UX™ System* is stored in the primary release area, **/srv/release/PRIMARY**, after installation. You must set up secondary releases explicitly in **/srv/release/***release-name*, where *release-name* is the name of the secondary release, such as **dgux_542**. For a picture of the primary and secondary release file structures, see Chapter 14.

When you have created a secondary release area following the instructions in Chapter 14, adding OS clients to the primary and secondary release areas is identical. You can associate an OS client with only one release at a time.

The examples in this chapter emphasize the addition of OS clients to the primary release because they are the most common. If you are adding clients to a secondary release, you can substitute the current *release-name* for each instance of PRIMARY in the pathnames given in this chapter.

Adding a diskless OS client

Perform these procedures at the OS server:

- Complete the planning worksheets.
- Add an OS client to the network databases.
- Build a first-time custom kernel for the client.
- Add an OS client to the operating system release.

- Boot the OS client kernel.
- Set up packages.

These procedures are discussed below.

Network planning for OS clients

Two OS client network planning worksheets follow this section to help you prepare for installation. They list network parameters for which you need to supply values. Completing the worksheets before you begin the installation procedures for OS clients will speed up the installation process considerably.

If you have experience installing the DG/UX system or other operating systems, you may prefer to go directly to the OS client network planning worksheets. The following list defines the network parameters you will need to complete the planning worksheets.

Hostname	A unique name you assign to your computer system functioning as an OS client that can contain up to 31 alphabetic and numeric characters. However, you are advised to keep the names short. Host names that relate to the use or location of the system are particularly helpful in networked environments where hosts may share file systems. Examples of hostnames are **fred**, **jamaica**, and **writer_doc**. Do not use the capitalized names **MY_HOST** or **PRIMARY**; these names are reserved by the system.
Internet address	You provide the Internet address of the host being set up. An example of an Internet address is **128.223.2.1**. For details, see *Managing TCP/IP on the DG/UX™ System*.
Ethernet address	Host address that is unique to the particular hardware. The factory sets this address. It consists of six 2-digit (hexadecimal) fields separated by colons in the form *nn:nn:nn:nn:nn:nn*.
NIS domain name	A named set of NIS maps located in **/etc/yp/***domain-name*. Computer systems having this directory as their default NIS domain share the data found in its maps.
Do you subnet?	Subnetting allows you to view multiple physical networks as a single logical

	network. You must answer "yes" or "no" to the query about subnetting.
Network mask	A hexadecimal bit pattern that specifies the number of bits used to identify the network part of an Internet address. **0xff000000** and **0xffff0000** are examples of network masks. You need to specify a network mask only when your network is subnetted.

Planning worksheets

Figure 18–4 provides a worksheet in which you record the Internet address and Ethernet address for each OS client on your system; Figure 18–5 is another worksheet for entering the appropriate ONC and TCP/IP parameter values required for each OS client's package setup.

OS Client Hostname	Internet Address	Ethernet Address
Example: junior	*123.227.3.14 (use periods)*	*08:00:1B:03:45:11 (use colons)*

Figure 18–4 OS client network planning worksheet (Part 1)

ONC and TCP/IP Parameters	Example Values	Actual Values
NIS domain name	*my_domain*	
Do you subnet?	*yes*	
Network mask	*0xffffff00*	

Figure 18–5 OS client network planning worksheet (Part 2)

Adding OS clients to the OS server's network databases

To add an OS client to a number of network databases, you perform the following tasks at the OS server:

- Add the OS client to the OS server's **hosts** database.
- Add the OS client to the OS server's **ethers** database.
- Add users on the OS client to the OS server' **host.equiv** database.

IMPORTANT: If you installed the ONC/NFS package on the OS server and you declared the OS server an NIS server (discussed in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*), then you must set up OS clients in the NIS **hosts** and **ethers** databases instead of local **hosts** and **ethers** databases. However, you must delay the addition of OS clients to a release until the additions of OS clients to the NIS **hosts** and **ethers** databases have time to propagate through the network. Therefore, you should add OS clients to the NIS **hosts** and **ethers** databases well before (for example, the day before) adding OS clients to the operating system release.

Adding an OS client to the OS server's hosts database (/etc/hosts)

The **hosts** database contains a list of Internet address and OS client hostname pairs to enable network communications between the OS server and its OS clients. To add an entry to the **hosts** database, perform the procedures in this section at the OS server.

1. Follow this path through **sysadm** and respond to the prompts:

```
Networking-> TCP/IP-> Databases-> Hosts-> Add
```

The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays

```
Hosts database to use: [NIS (YP)]
```

If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Hosts database will be used.` **Sysadm** will let you insert a new entry in the *local* **hosts** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

```
Hosts database to use: [NIS (YP)] ↵
```

3. **Sysadm** displays:

```
Host Name:
```

4. Consult the OS client network planning worksheet that you completed and enter the hostname; for example:

```
Host Name: junior ↵
Internet Address:
```

5. From your OS client network planning worksheet, enter the four-field (four octet) Internet address. For example:

```
Internet Address: 123.227.2.14 ↵
Alias List:
```

6. You may specify an alternate name (alias) for your host. For example, if your department uses long hostnames, you may choose a shorter alias for convenience. For example:

```
Alias List: jr ↵
OK to perform operation? [yes]
```

7. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example:

```
OK to perform operation?  [yes] ↵
junior has been added.
```

Sysadm has added the OS client to the appropriate hosts file. Repeat this procedure to add as many OS clients as desired to the **hosts** database.

Adding an OS client to the OS server's Ethernet database (/etc/ethers)

The **ethers** database contains a list of Ethernet address and OS client hostname pairs to enable network communications between the OS server and its OS clients. To add an entry to the **ethers** database, perform these steps at the OS server.

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

```
Networking-> TCP/IP-> Databases-> Ethers-> Add
```

The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS

server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays:

```
Ethers database to use: [NIS (YP)]
```

If your host is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Ethers database will be used.` **Sysadm** will let you insert a new entry in the *local* **ethers** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

```
Hosts database to use: [NIS (YP)] ↵
```

3. **Sysadm** prompts:

```
Host Name:
```

4. Consult the OS client network planning worksheet that you completed and enter the hostname; for example:

```
Host Name: junior ↵
Internet Address:
```

5. From your OS client network planning worksheet, enter the Internet address, form *hh:hh:hh:hh:hh* where *h* is a hexadecimal number. For example:

```
Internet Address: 08:00:1B:03:32:43 ↵
OK to perform operation?  [yes]
```

6. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example:

```
OK to perform operation?  [yes] ↵
junior has been added.
```

Sysadm has added the OS client to the appropriate **ethers** file. To add another OS client to the **ethers** database, repeat these steps.

Adding an OS client user to the OS server's trusted hosts database (/etc/host.equiv)

Allowing a user to access the server using a remote login may be helpful for such tasks as checking the print queue for a printer on the OS server. To add a user login name to the trusted **hosts** database, perform the procedures in this section at the OS server.

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

   ```
   Networking-> TCP/IP-> Databases-> Trusted Hosts-> Add
   ```

   Sysadm prompts:

   ```
   Host Name:
   ```

2. Enter the hostname to which the user will make remote requests and remotely log in. For example:

   ```
   Host Name: junior ↵
   User Name: [all]
   ```

3. Specify the login name of the user who will have remote request and login privileges. Or you can specify all (local) users. For example:

   ```
   User Name: [all] johnson ↵
   OK to perform operation?  [yes]
   ```

4. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example:

   ```
   OK to perform operation?  [yes] ↵
   johnson on junior has been added.
   ```

   The named user on the named remote system (the OS client) in the **/etc/host.equiv** file can access the OS server system using the remote shell (**rsh** or **remsh**) or the remote login (**rlogin**). Refer to the **rsh**(1), **remsh**(1), and **rlogin**(1) manual pages for more information.

Building a first-time custom kernel for an OS client

OS clients as well as the OS server must have kernels to provide operating system services. (Refer to Chapter 15 for information on kernels.) Perform the procedures in this section at the OS server.

The OS server must build each OS client's first kernel to establish an OS client's basic operation. You may build one kernel for all OS clients, or you may build individual kernels for each OS client. After an OS client is operational, it can build its own kernel if necessary.

Deciding whether to build a common or unique kernel

If you are adding multiple OS clients, decide whether to build a common kernel for all OS clients or individual kernels for each OS client. If all OS clients have identical hardware and software

configurations (for example, none has an attached I/O device), you can build a common kernel for all OS clients to share.

A primary advantage of a common kernel is to save mass-storage disk space. Conversely, a disadvantage is reduced security. Because all OS clients have root access to the kernel, any user with superuser privileges on an OS client computer can alter the kernel and change the common kernel.

The standard location for all kernels of the primary release of the DG/UX system is **/srv/release/PRIMARY/root/_Kernels**. Each OS client has a link named **dgux** in its primary release directory (**/usr/release/PRIMARY/root/***client-hostname*) to its kernel in the **_Kernels** directory. Multiple OS clients sharing the same kernel are linked to a common kernel from their client root directories. By convention, the common kernel is named **dgux.diskless**.

We recommend a unique kernel if an OS client's hardware configuration differs from the common OS clients' (for example, if the OS client has an attached parallel line printer). If the kernel includes an I/O device that is not physically attached to a given OS client computer, you can still boot that kernel, but an error message will appear indicating that the device could not be configured. A disadvantage of each unique kernel is that it requires more megabytes of disk space on the server.

Generally, you should build the most common kernel first, and then build each individual kernel (as needed). To create an individual kernel for an OS client, be sure to supply a unique system configuration filename.

1. To build a first-time kernel for an OS client, follow this path through **sysadm**:

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

   Sysadm prompts:

   ```
   System configuration file name: [xxx]
   ```

2. Enter the system configuration filename extension. A conventional name for a system file for a shared kernel is **system.diskless**. Entering **diskless** here yields the system filename **system.diskless**. For example:

   ```
   System configuration file name: [xxx] diskless ↵
   [system.diskless] Correct? [yes]
   ```

3. If you are happy with the name, confirm with Enter. Otherwise, enter **n** and enter a different name. For example:

```
[system.diskless] Correct? [yes] ↵
Build for this host or for OS client(s) of
     this host: [this host]
```

4. You want to build this for an OS client, so enter **OS Client** in either upper- or lower-case:

```
Build for this host or for OS client(s) of
     this host: [this host] OS client ↵
Editor: [/usr/bin/vi] ↵
```

We suggest you use the **vi** editor (the default) to edit the system file. Chapter 4 contains a **vi** command summary. You can specify another editor by entering its complete pathname at the prompt.

Sysadm now runs the editor you chose on the system file. The system file is long; it spans several screens. You should comment out configuration variables that do not apply to your system. Place a # in the first column of a line you want commented out. Read the contents of the system file before you edit it.

The three sections in the system file that you are most concerned about are:

- OS client configuration variables
- General configuration variables
- OS client hardware devices

OS client configuration variables

The system file section on OS client configuration variables follows.

```
# OS Client Configuration Variables:
#
# These configuration variables specify that the root and swap
# file systems will be mounted over NFS.
#

      NETBOOTDEV          "inen()"
      ROOTFSTYPE          NETWORK_ROOT
      SWAPDEVTYPE         NETWORK_SWAP
```

NETBOOTDEV, ROOTFSTYPE, and SWAPDEVTYPE provide resources to the OS client over the network. Make sure no comment symbols (#) appear in the first column of the lines containing these variables.

General configuration variables

The system file section on general configuration variables follows.

```
# General Configuration Variables:
#
# The NODE variable controls your nodename for uname(1) and
# uucp(1).
#
      NODE              "diskless"
```

NODE refers to the hostname of your computer. After you supply the name of the system file, **diskless**, it becomes the NODE name. However, each time the OS client is rebooted, this value is overridden by the hostname provided during TCP/IP package setup (covered in Chapter 13). The node name is used by the **uname** command to report the name and other attributes of the current system. Also, the UUCP facility uses the node name for performing file transfers on UNIX systems. It is also presented as part of the log-in banner message when you log in to your system. The node name is restricted to 31 characters.

The master files located in the **/usr/etc/master.d** directory contain complete lists of general configuration variables and default values. You accept the master file defaults by not resetting any variables in your system file. If you wish to override a master file default, reset it in your system file.

OS client hardware devices

Two sections of the system file list I/O devices that can be attached to an OS client. The first section relates to common hardware devices attached to an OS client. In the sample that follows, the I/O device names are not commented out.

```
# OS Client Hardware Devices:
#
# These hardware devices are found on most operating system
# clients. You must add any other hardware devices found on
# the client (using the examples below as a guide), and delete
# any devices which are not found on the client.
#
      kbd()      # -- keyboard
      grfx()     # -- graphics display
      inen()     # -- integrated Ethernet controller
      duart()    # -- integrated Duart terminal line controller
```

A second section offers more I/O devices that can be attached to the OS client. These I/O device names are commented out. Remove the pound sign (#) from the first line of those devices that reflect the OS client's configuration. As a convenience, the SCSI disk and tape

device names contain an asterisk in the SCSI ID field, which is a metacharacter that matches any SCSI ID value on any attached device. The section on typical OS client devices follows.

```
# Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both. Determine which situation applies.

#       kbd()          # -- keyboard
#       grfx()         # -- graphics display
#       sd(insc(),*)   # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)   # -- all SCSI tapes on integrated SCSI adapter
#       inen()         # -- integrated Ethernet controller
#       duart()        # -- integrated Duart terminal line controller
#       duart(1)       # -- second Duart (if present on system)
#       lp())          # -- integrated printer controller (if present)
```

If you have just loaded a new package on your system, check its release notice for information on possible variable tuning. If you need to tune a variable, either enter or modify it in the appropriate location in the system file.

A common source of kernel build failures is the absence of a comment symbol (#) used to flag notes to be ignored. Be sure you comment out all text you want ignored. Check your spelling and verify the DG/UX device names.

5. After you finish editing the system file, write the file and quit the editor by typing the **ZZ** command.

   The prompts that appear during a kernel build depend on whether your system has existing OS clients. The following prompt appears if your system has no existing OS clients:

   ```
   There are no operating system clients of this machine
   for which to link the new kernel. The new kernel may be
   used by any clients added later. The kernel pathname will
   be /srv/release/PRIMARY/root/_Kernels/dgux.diskless.
   Continue with the build?  [yes]
   ```

   The following prompt appears if your system *has* existing OS clients.

   ```
   Link the new kernel to which OS clients: [all] ↵
   ```

6. If you see the "`Continue`" prompt, confirm by pressing Enter; for example:

   ```
   Continue with the build?  [yes] ↵
   Configuring system...
   Building kernel...
   Successfully built dgux.diskless.
   ```

   Skip to step 10.

7. If you see the "`Link...to clients`" prompt, this means your system has OS clients. Continue with this step. **Sysadm** is asking

   ```
   Link the new kernel to which OS clients: [all]
   ```

   To boot a kernel, a client must be able to resolve the filename **dgux** in the release root directory of the client (**/usr/release/PRIMARY/root/***client-hostname*) to a kernel. **Sysadm** handles this by creating a link from filename **dgux** in each client's release root directory to the kernel name (in **/usr/release/PRIMARY/_Kernels**).

   If all (or most) of your clients can use the same kernel, you can save a lot of disk space by having links created in each client's release root directory to a single kernel in the **_Kernels** directory. This lets you use one kernel for multiple clients. To have **sysadm** create these links, select **all**, the default.

   If you're creating an individual kernel for a client that has special needs, you don't want to link for all clients to that kernel. To link a specific OS client to this kernel, enter the hostname of the OS client instead. If **sysadm** is not displaying a list of previously added OS clients (as with ASCII **sysadm**), enter **?** for a list.

   Whatever selection you make, the link replaces any existing links. If you select **all,** the program links all OS clients to the kernel. If you enter a client hostname, the program replaces any existing link only for that client; other existing client links are retained. This means, as mentioned earlier, that you should build the most common kernel first, have it linked for all clients, and then if needed create any individual kernels and have those linked for individual clients.

   Decide on your response and enter it. For example:

   ```
   Link the new kernel to which OS clients: [all] ↵
   ```

   After you answer this prompt, **sysadm** asks:

   ```
   Save the old kernel? [yes]
   ```

8. If you can afford the disk space, we recommend you accept the **yes** default response.

   ```
   Save the old kernel? [yes] ↵
   Continue with the build? [yes]
   ```

9. To continue the build process, press Enter:

   ```
   Continue with the build? [yes] ↵
   Configuring system...
   Building kernel...
   Successfully built dgux.temp
   ```

10. After the new kernel is built, the bootable kernel file is in the **/srv/release/PRIMARY/root/_Kernels** directory as **dgux.***name* and, if you so specified, it is also linked to **dgux** for the specified OS client(s) in each of their client root directories.

    The new kernel does not become effective until someone boots it over the network from an OS client. Do not boot the kernel yet. At the OS server, you must first add each OS client to the DG/UX system release as explained in "Adding an OS client to a release."

Adding an OS client to a release

Perform these procedures at the OS server.

After the OS server is operational — that is, a custom DG/UX system is running in the server's root, and a DG/UX release has been installed and a kernel built in the primary release area (and perhaps secondary release area) — then you can attach each OS client to the release it wants. You identify a release to an OS client by the name of its release area, the name of the OS server; the pathname of the OS client's home directory, the OS client's swap size, and the pathname of the OS client's bootable kernel and bootstrap. If you have several clients to add, you can create a default client set with the values you want and use them as a base for each client you add.

It may seem that a server running the one DG/UX release cannot support a client with an attached disk with that uses another DG/UX release. However, such a server *can* support such a client, and vice-versa. This is true because the systems are communicating over the network, not using the same physical disk, and disk format restrictions between the releases do not apply.

To add several OS clients to a release, you will perform the following tasks:

- Create a default OS client set.
- List the contents of a default OS client set.
- Modify a default OS client set.
- Add each OS client (based on the values in the default OS client set).
- List the added OS clients.

To add a single OS client, just add the OS client; you can skip the client set tasks.

If you installed the ONC/NFS package on your system and you declared the OS server an NIS master, you set up OS clients in NIS

hosts and **ethers** databases instead of local hosts and **ethers** databases. You must delay the addition of OS clients to an operating system release so that the additions of OS clients to the NIS hosts and **ethers** databases have time to propagate through the network. Therefore, you should add OS clients to the NIS hosts and **ethers** databases at least several hours (perhaps a day) before adding OS clients to the release. If you do not wait long enough, addition of OS clients to the operating system release will fail. See "Adding OS clients to the OS server's network databases" for more information.

▶ To add an OS client to a release:

1. If you wish to define a set of default values for the queries in the Add operation, use the `Client-> OS Client-> Defaults-> Create` operation. Use `Client -> OS Client-> Defaults-> Set` operation to make the defaults set the current set.

2. Add the client's Internet address to the **hosts** database with the operation `Networking-> TCP/IP-> Databases Hosts-> Add`.

   Get the Internet address from someone who manages the client system. Then add the client's Ethernet address to the **ethers** database with the operation `Networking-> TCP/IP-> Databases-> Ethers-> Add`.

   Some AViiON computers display their Ethernet address when you turn on power. See the Networking menu for more information on the **sysadm** operations for adding network database entries. For more information on the networking packages, see *Managing TCP/IP on the DG/UX™ System* and *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

3. Make sure the following directories have sufficient space:

   /srv/swap
   This directory is for the files that OS clients use as their swap areas. The file system containing this directory must be large enough to hold the swap areas for *all* OS clients that your system serves.

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

```
(number-of-OS-clients * blocks-per-OS-client) + 7%
overhead
```

```
(4 * 50,000) + 7% = 200,000 + 7% = 214,000 blocks
```

If your swap space is not sufficient once your system is running, use the free swap value reported by the **sar** command to determine the available swap space. (Divide the amount of free swap space by 2048 to convert blocks to megabytes.) You should maintain the amount of free swap space at 15% to 30% of the total physical memory plus swap area space on the system. For more information on calculating OS client swap space, see "OS client swap space (srv_swap)."

/srv/dump

This directory is for memory dump images that OS clients may produce in the event of a system panic or any time someone makes a memory dump of the client. The file system containing this directory should be large enough to hold at least one system memory image of an OS client. A system's memory dump is the same size as its physical memory, or, if taking only a kernel dump rather than a complete dump, about half that size. This directory is optional but recommended.

/srv/release/*release***/root**

This directory is for the root directories of the OS clients. You need one such directory for each release supported by your OS server. A DG/UX system root directory requires 40,000 512-byte blocks. The file system containing this directory must be large enough to contain the root directories for all OS clients attached to the release.

/srv/release/*release***/usr**

This directory is for the host-independent (**/usr**-equivalent) directory of a secondary release. This directory is intended to contain the portion of an operating system that does not vary from system to system within that release. You do not need this directory if all of your OS clients use the PRIMARY release. You need one such directory for each secondary release supported by your OS server. The file system containing this directory needs to be large enough to hold the host-independent software for the release.

If the file systems that will contain these directories are not large enough, you may need to increase the size of the file systems or create new file systems. For information on creating virtual disks, adding a file system to the **/etc/fstab** file, and expanding the size of a file system, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

4. Perform the Client–> OS Client–> Add operation for each client as explained in "The sysadm Add operation." Note that you can attach a client to more than one release by performing the Add operation once for each release.

5. Set up software packages on the client system.

IMPORTANT: Because of basic operating system differences, the **sysadm** operations 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 or your Data General representative.

The sysadm Add operation

The **sysadm** operation `Client-> OS Client-> Add` requires that you supply the following information:

`Client Host Name`
: This is the name by which the OS client will be known on a network. You should have already added **hosts** and **ethers** entries for the client in your TCP/IP databases.

`Release Area`
: This is the operating system release for the OS client. You should have already added this release using operations in the Software menu. Adding a client to a release does not set the client to boot that release if the client is already set to boot a different release. You set a client's current release with the Set operation. The default release, **PRIMARY**, already exists because it is the currently installed release of the DG/UX system on the server. Make sure the file system containing the release has enough free space for the new root directory. A DG/UX system root requires 20 megabytes (40,000 512-byte blocks) by default.

`Server Host Name`
: This is the name of the system that will be the client's OS server. The reason you have a choice here is because systems with more than one network device have more than one host name. A system connected to two different networks, for instance, has one host name for one network interface and another host name for the second network interface. You should select the host name for the network interface connected to the client's network.

`Home Directory`
: This directory is the one that will contain the home directories of users on the client. On the system where the directory resides, the directory must be exported. The Add operation modifies the client's file system table (**/etc/fstab**) to make the directory available on the client system.

`Swap Size`
: This is the amount of swap space that you want to allocate

for the client system. There is no formula for calculating the ideal amount of swap space for a system. A general rule is to allocate 1.5 times the amount of physical memory in the client computer. The default physical memory size, 16 Mbytes, is about 32,000 512-byte blocks. Multiplying 1.5 by 32,000 blocks produces 48,000 blocks, which rounds up to 50,000 blocks. Make sure the file system containing the **/srv/swap** directory has enough free space to hold the swap file.

`Kernel Pathname`

This is the pathname of the kernel image that the client will use. OS client kernels generally reside in **/srv/release/***release_name***/root/_Kernels**.

`Bootstrap File`

This is the pathname of the bootstrap file that the client will use. The default, **/usr/stand/boot.aviion**, is for the PRIMARY release.

Files created by the Add operation

The Add operation copies or modifies a number of files. This section outlines some of these changes.

When you add a client, the client inherits the server's environment. This means that the client receives information from the server's various parameter files: **dgux.params**, **tcpip.params**, **nfs.params**, and so on. Clients can modify these as they wish. The Add operation 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 OS client kernel, a link to a common OS 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

An OS client's root space is created on the server. For the **PRIMARY** release, **sysadm** copies the files in **/srv/release/PRIMARY/usr/root.proto** to the client root area. The client root area is **/srv/release/***release_name***/root/***client*, where *release_name* is the name of the release and *client* is the client's host name.

Make sure the file system containing **/srv/release/***release_name***/root/***client* has enough free space to hold the client root. The root directory for a DG/UX system requires 20 Mbytes (40,000 512-byte blocks) by default.

Client swap file

The client swap file is **/srv/swap/***client*. Make sure the file system containing **/srv/swap** has enough free space to hold the swap file for all OS clients. The swap file is fixed at the size you define.

Client fstab file

The Add operation adds the following entries to the client's **/etc/fstab** file, whose pathname is **/srv/release/***release_name***/root/***client***/etc/fstab** on the server:

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

where *server* is the server's host name, *client* is the client's host name, and *home_dir* is the file system containing the home directory of the client's users.

The Add operation 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).

Client nfs.params file

The Add operation modifies the client's **/etc/nfs.params** file, whose pathname is **/srv/release/***release_name***/root/***client***/etc/nfs.params** on the server, and sets the NIS (Network Information System) domain name to be the same as the server's NIS domain name. For information on NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

Client kernel link

The **/srv/release/***release_name***/root/_Kernels/dgux.diskless** file is booted by an OS client by default. The Add operation links this file, if it exists, to a file in the client's root directory, **dgux**. This kernel does not exist on the DG/UX system as shipped. If you intend for your OS clients to use this kernel, you need to build it. Use the operation `System-> Kernel-> Build`, discussed in Chapter 15.

Server bootstrap link

An OS client uses a secondary bootstrap to load **dgux.diskless** over the network. This default bootstrap file is **/usr/stand/boot.aviion**.

The Add operation makes a symbolic link from **/usr/stand/boot.aviion** to **/tftpboot/***client_ip_addr*, where *client_ip_addr* is the client's Internet address in hexadecimal. The Add operation gets the client's Internet address from the **/etc/hosts** file.

Server bootparams file

The Add operation puts the following entry in **/etc/bootparams** for an OS client on an OS server:

```
client  root=server:/srv/release/PRIMARY/root/client \
        swap=server:/srv/swap/client \
        dump=server:/srv/dump/client
```

where *client* is the client host name, and *server* is the server host name.

> The entry is actually a single logical line. In the 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 Add operation puts the following entries in **/etc/exports** for an OS client on an OS server:

```
/srv/release/PRIMARY/root/client  -access=client,root=client
/srv/swap/client  -access=client,root=client
/srv/dump/client  -access=client,root=client
```

where *client* is the client host name.

If the **/usr** file system is not already exported, the operation adds it to the server's **/etc/exports** file.

Client and release data files for sysadm

The Add operation 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 an OS client from a release

Select the **sysadm** operation `Client-> OS Client-> Delete` to remove a client from a release.

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 operation displays each client/release pair and asks if it should be deleted.

The Delete operation prompts you for the name of the release. Optionally, you may choose to save the client's root file system.

Modifying an OS client's bootstrap link

Select the `Client-> OS Client-> Modify` operation to change the pathname of the file used as a client's secondary bootstrap. The Modify operation prompts you for the client name, the release name, and for the new bootstrap file pathname.

Listing OS client information

To verify the successful addition of OS clients, follow this path through **sysadm** and respond to the prompts:

```
Client-> OS Client-> List
```

Specify a client hostname or **all**. If you specify all clients, a list of name and release areas appears. For example:

```
Client Host Name: all ↵

OK to perform operation?  [yes] ↵

Client Name             Release Area
-----------             ------------
matilda                 PRIMARY
junior                  PRIMARY
client_dgux542          dgux542
```

Changing an OS client's boot release

In the case where a client is attached to more than one release, the `Client-> OS Client-> Set` operation allows you to change the default boot path. Adding a client to a new release with the `Client-> OS Client-> Add` operation does not change the client's default boot path; you must change the default boot path explicitly with the Set operation.

Managing OS client defaults sets

The **sysadm** operation `Client-> OS Clients-> Defaults` provides operations for managing sets of default values that appear when you use the Add operation to add a client to a release. The menu provides operations to create, remove, modify, and list defaults sets. There is also an operation to select which defaults set is currently used by the Add operation.

Creating a defaults set

A defaults set specifies values to be shared by OS clients linked to a common kernel. The defaults set provides default values for each client you add. Creating your own defaults set can save you a lot of time as you add OS clients.

A defaults set named **unnamed** is supplied with DG/UX, but you may want to create a custom set with a different name. To create a defaults set, perform these steps at the OS server:

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

   ```
   Client-> OS Client-> Defaults-> Create
   ```

   Sysadm prompts:

   ```
   Set Name: [unnamed]
   ```

2. You can either accept the defaults set name, **unnamed**, or you can create a client set to be customized. Each defaults set name must be unique, so if the set **unnamed** has already been created, you will need to use another name. To accept the default name, **unnamed**, and its defined attributes, press Enter; to create a new defaults set, enter its name. For example, to create a defaults set named **dgset**:

   ```
   Set Name: [unnamed] dgset ↵
   OK to perform operation?  [yes]
   ```

3. To create the new set, press Enter:

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

   You have created a defaults set.

Modifying a defaults set

Use the **sysadm** operation `Client-> OS Client-> Defaults-> Modify` to set or change the values in an OS client defaults set created with the Create operation. A defaults set includes the same parameters that you see in the Add operation of the OS Clients menu.

The parameters you need to specify to modify a defaults set are:

- Set name
- Release area
- Server host name
- Home directory
- Swap size
- Kernel pathname
- Bootstrap file

See the section on the OS Client menu's Add operation for a discussion of each parameter. Modifying a defaults set has no effect on OS clients added when the defaults set was in use.

After creating a custom defaults set, you must modify it. If you accept the predefined defaults set named **unnamed**, you do not need to modify it.

1. To modify the defaults set, follow this path through **sysadm**:

   ```
   Client-> OS Client-> Defaults-> Modify
   ```

   Sysadm prompts:

   ```
   Set Name: [default]
   ```

2. Enter the name of the defaults set you created with the Create function. For example, **dgset**:

   ```
   Set Name: [default] dgset ↵
   Release Area: [PRIMARY]
   ```

3. The default release name is PRIMARY. If you want to specify a different release area, that release area must exist and you must enter its full pathname. To select the primary area, press Enter:

   ```
   Release Area: [PRIMARY] ↵
   Server Host Name: [xxx]
   ```

4. The default server hostname, *xxx*, is your current hostname. Generally, since you are running **sysadm** on the server, this default is correct. If so, take the default. For example:

   ```
   Server Host Name: [boss] ↵
   Home Directory: [/home]
   ```

5. This sets the name of the file system on the server that will contain the home directories for the users on the OS client system. (Someone on the client will need to add these users via with the **sysadm** `User-> Login account` operation.) **Sysadm** will create

a mount point for the file system relative to the OS client; then it will add the directory name to the OS client's file system table **/etc/fstab** so it will be mounted automatically at client startup. The file system you specify must already be mounted on the OS server and be exportable; that is, someone on the server must have added the directory as a file system and made it exportable. The file system directory should be the same as the parent directory you specify when adding a user on the OS client.

If you do not specify a directory, **sysadm** does not add a home directory entry to the OS client's **/etc/fstab** file. To not specify a directory in the ASCII **sysadm**, enter **none**; in the X window version, erase the default name.

For example, to select the default file system, /home:

```
Home Directory: [/home] ↵
Swap Size (in megabytes): (4-128) [24]
```

6. For the swap size, accept the default or enter the size of **/srv/swap** you recorded in your planning worksheet for virtual disks.

```
Swap Size (in megabytes): (4-128) [24] ↵
Kernel Pathname:
  [/srv/release/PRIMARY/root/_Kernels/dgux.diskless]
```

7. This specifies the pathname of the kernel that all clients will use. (**Sysadm** will create a link in each client's release directory to this pathname.) You created this kernel in a previous section. If you used the default kernel name, diskless, then you can take the default. If you specified a different kernel name, specify the full pathname (**/srv/release/PRIMARY/root/_Kernels/dgux.***xxx*), where *xxx* is the name you gave). If you created a special kernel for any client that has unique needs, you can modify this value later for that client using the **sysadm** Client menu. For example, for the default:

```
Kernel Pathname:
  [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵
Bootstrap File: [/usr/stand/boot.aviion]
```

8. The bootstrap file contains the bootstrap needed by each OS client to boot a kernel. Accept the default bootstrap pathname:

```
Bootstrap File: [/usr/stand/boot.aviion] ↵
OK to perform operation? [yes]
```

9. Press Enter to accept the values you have entered.

```
OK to perform operation? [yes] ↵
Defaults for set dgset have been modified.
```

You have finished the modify/add operation. Next, if you have not already done so, you should make your custom set the default.

Listing defaults sets

An OS client defaults set specifies general attributes of the operating system environment shared by all OS clients linked to a common kernel. To list the attributes of a defaults set, perform the procedures in this section at the OS server.

Use the **sysadm** Client–> OS client–> Defaults–> List operation to display the parameters for an OS client defaults set. See the section on the OS Client menu's Add operation for a discussion of each parameter.

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

   ```
   Client-> OS Client-> Defaults-> List
   ```

   Sysadm prompts:

   ```
   Set Name: [default]
   ```

2. Enter a name or ? for a list of names:

   ```
   Set Name: [default] dgset ↵
   ```

3. Confirm with Enter and **sysadm** will list parameters for the set:

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

   Parameter          Value
   ---------          -----
   client_release     PRIMARY
   client_server
   client_homedir     /home
   client_swapsize    24
   client_kernel
   /srv/release/PRIMARY/root/_Kernels/dgux.diskless
   client_bootstrap /usr/stand/boot.aviion
   ```

   The values displayed for your defaults set may differ.

Selecting a defaults set

Use the `Select` operation to set the current OS clients defaults set to be used by the Add operation. Adding an OS client defaults set does not automatically make that set the current one; you need to use the Select operation to make it current.

1. To select a default client set, follow this path through **sysadm**:

   ```
   Client-> OS Client-> Defaults-> Select
   ```

   Sysadm prompts:

   ```
   Set Name: [system]
   ```

2. Specify the name you want as the system default. If the default set names are not on display, enter a question mark (?) for a list. For example:

   ```
   Set Name: [system] ? ↵

   ...Choices are

        1  system
        2  dgset
        3  unnamed
   ...
   Set Name: [system]
   ```

3. Specify the name or menu number of the set you want to be the default set. For example:

   ```
   Set Name: [system] dgset ↵
   OK to perform operation? [yes]
   ```

4. Confirm your choice:

   ```
   OK to perform operation? [yes] ↵
   Default set dgset has been selected.
   ```

Removing a defaults set

Use the **sysadm** operation `Client-> OS Client-> Defaults-> Remove` to delete an OS client defaults set created with the Create operation. Deleting a defaults set has no effect on clients added using the set.

Booting an OS client kernel

A kernel must be running on an OS client before anyone can set up packages and log in that client. Perform these procedures on the OS client.

1. If the client you want to boot has a DG/UX system running, you must first shut down the system to reboot. If it does not have DG/UX running, skip to step 2.

   Use these commands to shut down the client system:

```
# cd / ↵
# shutdown -g0 -y ↵
# halt ↵
SCM>
```

Your screen should display the `SCM>` prompt.

2. Boot the OS client's kernel over the network to a run level of i, using the command form **b** *network-controller* **–i**, where the *network-controller* specifies the server's network controller connected to this client, controller type **dgen** or **inen**. For example:

   `SCM>` **b dgen(0) –i** ↵

   Your screen displays a series of booting messages during which you must confirm the current date and time. You must also specify the boot device. The first set of booting messages looks like the following:

```
Booting inen(0) -i
DG/UX System 5.4R3.10 Bootstrap
Loading image ........

Network boot device [?]
```

3. Enter the same network controller name you specified in step above; for example:

```
Network boot device [?] dgen(0) ↵
```

   DG/UX displays more boot messages as follows.

```
Linking short names for /dev device nodes .....
Starting disk daemons ....
Mounting local file systems .......
Starting installation steps ....

Set up package(s)? [yes]
```

   At this point, you need to set up software packages and mount file systems for the OS clients to access what your users need to do their work. For how to set up packages, see "Setting up packages on the OS client." For how to add local and remote file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

Setting up packages on the OS client

Before the OS client can be fully operational, someone must set up DG/UX packages.

Perform these procedures at the OS client. Use the OS client package setup worksheet you completed earlier when setting up packages.

When prompted to set up packages, request help by entering a question mark (?). Figure 18–6 shows a typical dialog for starting the package setup process; the dialog selects NFS, ONC, TCP/IP, and X11.

```
Package Name(s):  [all] ? <Enter>

Select the package(s) you want to set up. If you want to set up all
packages, select 'all.'  If you select 'all,' do not select any
individual package names.

Choices are

1 all
2 dgux
3 gcc
4 networker
5 nfs
6 onc
7 tcpip
8 X11
9 X11.sde
10 xdt
11 dgux.man
...
Enter a number, a name, ...q to quit: 5, 6, 7,8, 9 <Enter>
OK to perform operation? [yes] <Enter>
```

Figure 18–6 Typical OS client package setup dialog

The choices listed depend on the packages that were loaded and set up on the OS server.

Setting up NFS

The following messages appear for **NFS**:

```
Setting up nfs in MY_HOST root.

    Setting up NFS in MY_HOST root.............

    Creating NFS run level links..........

    Initializing NFS prototype files.........

    NOTE:  See
/srv/release/PRIMARY/root/MY_HOST/var/setup.d
           /log/nfs.root for a detailed account of the
root
           setup of NFS.

Package nfs has been successfully set up in MY_HOST root.
Package setup for nfs is complete.
```

Setting up ONC

In setting up **ONC**, you need to supply one value: the name of the NIS domain. Use the OS client package setup worksheet you completed earlier when setting up packages.

```
Setting up onc in MY_HOST root.

Setting up ONC in MY_HOST root..........

Creating ONC run level links............

Initializing ONC prototype files........

Enter the NIS Domain name []: work-net ↵
[work-net] Correct? [yes]: ↵

NOTE:  This host will first run as an NIS client.

NOTE:  See /srv/release/PRIMARY/root/MY_HOST/var/setup.d
       /log/onc.root for a detailed account of the root
       setup of ONC.

Package onc has been successfully set up in MY_HOST root.
Package setup for onc is complete.
```

IMPORTANT: If you are upgrading, the NIS domain name is supplied by default.

Setting up TCP/IP

Setup of TCP/IP requires more planning than other packages. You must supply configuration information to the **sysadm** prompts. Use the OS client package setup worksheet you completed earlier when setting up packages. Also, refer to *Managing TCP/IP on the DG/UX™ System* for information on setting up the TCP/IP package.

```
Setup package tcpip in MY_HOST root

       Setting up tcpip...

       The following queries refer to the primary
         network interface.

       Enter host name: junior ↵
       [junior] Correct? [yes] ↵
       Enter host Internet address: 128.222.8.60 ↵
       [128.222.8.60] Correct? [yes] ↵
       Is your local network subnetted? [no] yes ↵
       Enter the network mask: 0xffffff00 ↵
       [0xffffff00] Correct? [yes] ↵

       NOTE: Using inen0 as the primary network interface
                   controller.

       Package tcpip has been ... set up in MY_HOST root.

       NOTE:  See /var/setup.d/log/tcpip.root file for a
           verbose description of the package setup for
root.

       Package setup for tcpip is complete.
```

Setup for TCP/IP is complete.

Setting up X11

The following messages appear for **X11**:

```
Setting up X11 in MY_HOST root.
Package X11 has been ... set up in MY_HOST root.
Package setup for X11 is complete.
```

Setup for X11 is complete.

Bringing the OS client up to run level 3

Perform these procedures at the OS client.

After you set up packages, **sysadm** asks if you want to build and reboot another kernel; answer **no** as follows:

```
Build kernel? [yes] no ↵
Reboot now? [yes] no ↵
DG/UX Operating System       Release n

Console Login:
```

At the prompt, type the **sysadm** login name. The screen displays a shell prompt (#) to indicate a successful login. At the prompt, enter **init 3**, to move to run level 3, as follows:

```
# init 3 ↵
```

The screen displays a series of messages indicating a transition from run level **i** to **3**. A login prompt appears at the conclusion of the process. The type of login prompt depends on whether or not your computer system is equipped with the DG/UX X Window System package. For how to log in, see Chapter 2.

Adding an OS client that has a local root and swap virtual disk

This type of OS client has an attached disk drive that can provide its own local **root** and **swap** virtual disk OS resources (/ file system and swap space), while receiving its **usr** virtual disk OS resources (/**usr** file system) from the OS server via a LAN.

These procedures assume familiarity with the steps required to install a diskless OS client, and knowledge of **sysadm.** Before you begin these procedures, make sure that:

- The AViiON computer to be used as an OS server is running the DG/UX system (its kernel has booted to a run level of 3) and is connected to a network.
- You have created the required virtual disks to support OS clients and have mounted the required file systems. For how to create virtual disks and mount file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.
- The AViiON computer to be used as an OS client and its attached disk drive have been successfully powered up.

You will perform these steps:

- Complete the planning worksheet.
- Build a temporary OS client kernel.
- Add the OS client to the DG/UX release.
- Build an OS client kernel with local **root** and **swap** virtual disk resources.

- Load the / file system onto the OS client's local disk.
- Set up the software packages.
- Link the OS client's Internet address to the **/tftpboot** directory.

IMPORTANT: Examples illustrate the procedures documented in this section. Be sure that you provide the values that reflect your particular configuration.

Completing the planning worksheet

Figure 18–7 is a planning worksheet on which you should write information you will supply to **sysadm** prompts and system-level files when adding an OS client with a local **root** and **swap** virtual disk. You supply some items several times in different contexts.

Recording this data ahead of time accelerates the process. Not having this information on hand when **sysadm** requests it or when you need to type an entry in a system-level file will disrupt this procedure. Providing incorrect data may force you to abort the process and begin again.

Parameter Type	Example Value	Actual Value
OS client hostname	*junior*	
OS client Internet address*	*128.222.3.86*	
OS client Ethernet address†	*08:00:1B:18:03:11*	
OS client disk drive name	*sd(insc(0),0,0)*	
OS server hostname	*boss*	
OS server Internet address	*128.222.3.120*	
Temporary kernel name	*dgux.temp*	
NIS domain name	*my_domain*	
Do you subnet?	*yes*	
Network mask	*0xffffff00*	
OS client's Internet address (hexadecimal equivalent†)	*80DE0356*	

*The screen of the computer used as the OS client shows the Internet address in hexadecimal format the first time it boots its temporary kernel to a run level of 1. You can record its hexadecimal address at that point for later use. See the section "Loading the root (/) file system on the OS client's local disk" later for the procedure that produces that screen message.

†The Ethernet address appears when the system is powered on. You can record it then.

Figure 18–7 Local root and swap virtual disks configuration planning worksheet

Building a temporary OS client kernel

To create a temporary system file, perform these procedures at the OS server.

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

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

Sysadm prompts:

```
System configuration file name: [xxx]
```

2. Type **temp** and press Enter:

```
System configuration file name: [xxx] temp ↵
[system.temp] Correct? [yes]
```

3. **Sysadm** appends the name you type to the base filename **system** to yield the full name **system.temp**. Confirm with Enter:

```
[system.temp] Correct? [yes] ↵
Build for this host or for OS client(s) of this
      host: [this host]
```

4. Enter **OS client**:

```
Build for this host or for OS client(s) of this
host: [this host] OS Client ↵
Boot Device: [inen(0)]
```

5. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(*n*)**, where *n* is a number 0 through 5. For example:

```
Boot Device: [inen(0)]  dgen(0) ↵
Editor: [/usr/bin/vi]
```

6. Press Enter to accept the default editor, **/usr/bin/vi**.

```
Editor: [/usr/bin/vi] ↵
```

7. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the comment indicator # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section of the file follows, showing the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both. Determine which situation applies to your system.

#       kbd()  # -- keyboard
#       grfx()                # -- graphics display
        sd(insc(),*)          # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)          # -- all SCSI tapes on integrated SCSI adapter
        inen()                # -- integrated Ethernet controller
#       duart()               # -- integrated Duart terminal line controller
#       duart(1)              # -- second Duart (if present on system)
#       lp()                  # -- integrated printer controller (if present)
```

8. Save the file and quit the editor by typing **ZZ**.

   Sysadm prompts:

```
Link the new kernel for which clients:[all]
```

9. Do not link the new kernel to all clients. Instead, enter the OS client's name you gave when you created a diskless OS client earlier. For example, **junior**:

```
Link the new kernel for which clients:[all] junior ↵
```

Now, if a kernel with the name of this kernel already exists, **sysadm** will ask if you want to save the old kernel. Generally, to conserve disk space, we suggest you answer **no**. **Sysadm** prompts:

```
Continue with the build? [yes]
```

10. Press Enter to continue with the build:

```
Continue with the build? [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.temp.
```

You have successfully built a temporary kernel named **dgux.temp**.

Adding the OS client to the DG/UX release

Perform these procedures at the OS server.

You will next create, modify, and select a client default set, which provides the default parameters that appear at the system prompts as you add the OS client. Then you will add the OS client to the release. Refer to "Adding an OS client to a release."

Supply the following data from the planning worksheet to the prompts:

- OS server hostname
- Temporary kernel name
- OS client hostname

By completing these procedures, you add an OS client to the DG/UX release.

Building an OS client kernel with local root and swap virtual disk resources

Perform these procedures at the OS server.

1. Follow this path through **sysadm** to create a new system file:

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

Sysadm prompts:

```
System configuration file name: [temp]
```

2. Enter **local_root_swap**:

```
System configuration file name: [temp] local_root_swap↵
Build for this host or for OS client(s) of this
   host: [this host]
```

The name you type is appended to the supplied base name **system** to yield the full name **system.local_root_swap**.

3. Enter **OS client**:

```
Build for this host or for OS client(s) of this
host: [this host] OS Client ↵
Boot Device: [inen(0)]
```

4. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0).** Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(*n*)**, where *n* is a number 0 through 5. For example:

```
Boot Device: [inen(0)]   dgen(0) ↵
Editor: [/usr/bin/vi]
```

5. Press Enter to accept the default editor **/usr/bin/vi**:

```
Editor: [/usr/bin/vi] ↵
```

6. Search for the section in the file labeled "OS Client Configuration Variables." Comment out (insert a # in the first column position of) the lines containing the ROOTFSTYPE and SWAPDEVTYPE variables. At the end of the list, add the NETSTART variable and its assigned value **REAL_NET**. Be sure you type the variable name and value in uppercase letters. Use the Tab key for the desired alignment.

```
OS Client Configuration Variables

NETBOOTDEV          "inen()"
# ROOTFSTYPE         NETWORK_ROOT
# SWAPDEVTYPE        NETWORK_SWAP
NETSTART             REAL_NET
```

7. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section of the file follows, showing the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both. Determine which situation applies.

#       kbd()                 # -- keyboard
#       grfx()                # -- graphics display
        sd(insc(),*)          # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)          # -- all SCSI tapes on integrated SCSI adapter
        inen()                # -- integrated Ethernet controller
#       duart()               # -- integrated Duart terminal line controller
#       duart(1)              # -- second Duart (if present on system)
#       lp()                  # -- integrated printer controller (if present)
```

8. After you finish editing the file, save it and quit the editor (**ZZ**). After several messages, **sysadm** prompts:

   ```
   Link the new kernel for which clients: [all]
   ```

   IMPORTANT: Do not accept the default.

9. To the Link prompt, enter none:

   ```
   Link the new kernel for which clients: [all] none ↵
   Continue with the build? [yes]
   ```

10. Press Enter to accept the `yes` default response:

    ```
    Continue with the build? [yes] ↵
    ```

    You have successfully built a kernel named **dgux.local_root_swap**.

11. Quit **sysadm**.

12. From the shell, copy the **dgux.local_root_swap** kernel to **/usr/stand**:

```
# cp /srv/release/PRIMARY/root/_Kernels/dgux.local_root_swap  /usr/stand↵
```

13. Export the **/usr** file system and temporarily give root permission to the OS client (here **junior**) with the following shell command:

    ```
    # exportfs -iv -o root=junior  /usr ↵
    ```

    The **–i** option causes the entries in the **/etc/exports** file to be ignored. The **–v** option displays the name of the exported directory. The **–o** option assigns exclusive root access to OS client **junior.** See the **exportfs**(8) man page for more information.

IMPORTANT: Do not edit the **/etc/exports** file.

You have successfully built a kernel named **dgux.local_root_swap** that you will eventually boot from the OS client. This kernel provides local **root** and **swap** virtual disk resources. You have also exported the **/usr** file system (**usr** virtual disk resources) for the OS client's use.

Loading the root (/) file system on the OS client's local disk

Perform these procedures at the OS client.

1. Make sure the DG/UX system has been shut down. Your screen should display the SCM prompt. If it doesn't, shut down your system now using the following commands.

   ```
   # cd ↵
   # shutdown -g -y ↵
   # halt ↵
   SCM>
   ```

2. Boot the OS client's temporary kernel over the network to run level 1, typing the following command from the SCM prompt:

   SCM> **b inen(0) -1** ↵ (Or other network controller name, such as **dgen(0)**)

   Your screen displays booting messages as follows:

   ```
   Booting inen( ) -1
   Local Ethernet address is 08:00:1B:18:03
   Local Internet address is 128.222.3.86 or 80DE0356
   hex
   .


   temp
   DG/UX Operating System       Release n
   Console Login:
   ```

3. The booting message includes the OS client's Internet address in hexadecimal (in this example, `80DE0356 hex`). Write this address on your planning worksheet; you need it to complete the procedure.

4. Log in as **sysadm**:

   Console Login: **sysadm** ↵

   The screen displays a shell prompt (#) indicating a successful login.

5. Set the TERM variable using the following Bourne shell commands:

   ```
   # TERM=vt100 ↵
   # export TERM ↵
   ```

6. Invoke **sysadm**:

   ```
   # sysadm ↵
   ```

7. If the attached disk is new, make sure it is soft formatted and registered. For how to soft format and register a disk, see *Managing Mass Storage Devices and DG/UX™ File Systems*. The soft format automatically registers the disk. Then skip to step 10.

8. If the disk has been used for data storage, this means it uses logical disk format. You must convert it to virtual disk format. To do this, use the **sysadm** sequence `Device-> Disk-> Physical-> Convert.`

9. If the disk has been used for data storage, make sure the disk has 90,000 free blocks. To determine free disk space, use the **sysadm** sequence `Device-> Disk-> Physical-> List` and select the normal viewing style.

10. Create the **root** virtual disk, create a file system, and specify a length of 40,000 blocks.

11. Create the **swap** virtual disk with a length of 50,000 blocks. You do not need to create a file system this virtual disk.

12. Quit **sysadm**.

13. From the shell, make a directory and mount the local / file system at a temporary mount point using the following shell commands:

    ```
    # mkdir /mnt ↵
    # mkdir /mnt/tmp_root ↵
    # mount /dev/dsk/root /mnt/tmp_root ↵
    ```

14. Load the local / file system using the following commands:

    ```
    # cd /usr/root.proto ↵
    # tar –cf – . | (cd /mnt/tmp_root ; tar –xf –) ↵
    ```

    The copy process takes about one minute.

15. Copy the **dgux.local_root_swap** kernel into the OS client's local root directory using the following command:

    ```
    # cp /usr/stand/dgux.local_root_swap /mnt/tmp_root ↵
    ```

16. Create a **hosts** file using the following commands:

    ```
    # cd /mnt/tmp_root/etc ↵
    # cp hosts.proto hosts ↵
    ```

17. Using **vi**, open and edit the **hosts** file. Add one line with the Internet address (that you copied from the booting message in step 3) and OS server hostname to the file. Use the Tab key for the desired alignment. For example, for the OS server **boss** shown earlier in the sample worksheet in Figure 18–7, you would add:

    128.222.3.120 boss

    Type **ZZ** to save the file and quit the editor.

18. Create an **fstab** file using the following shell command:

    # cp fstab.proto fstab ↵

19. Using **vi**, open and edit the **fstab** file. First, comment out (insert the comment-indicating pound sign (#) in the first column of) the following line:

    ```
    # /dev/dsk/usr    /usr    dg/ux rw 1 0
    ```

    The preceding line specifies a locally mounted **/usr** file system.

    Next, insert this line to specify a remotely mounted **/usr** file system from the OS server. Use the Tab key for the desired alignment.

    boss:/usr /usr nfs bg,ro,hard x 0

    Type **ZZ** to save the file and quit the editor.

    You have successfully loaded the / file system on the OS client's locally attached disk, and created the correct entry to remote mount the **/usr** file system from the OS server.

Setting up packages

Perform these procedures at the OS client:

1. Halt the DG/UX system by typing this shell command:

   # **halt -q** ↵

2. Boot the new kernel to run level **i**, supplying the correct boot path at the SCM> prompt:

   SCM> **b sd(insc(0),0,0)root:/dgux.local_root_swap -i** ↵

   The screen displays a series of booting messages during which you must confirm the current date and time. Booting leads automatically to the DG/UX package setup process; press Enter when queried to begin package setup.

```
Booting sd(insc(),0)root:/dgux.local_root_swap -i
DG/UX System Release n Bootstrap
Loading image ........
.
Linking short names for /dev device nodes .....
Starting disk daemons ....
Mounting local file systems .......
Starting installation steps ....

Set up package(s)? [yes] ↵
Package Name(s): [all] ↵
OK to perform operation? [yes] ↵
```

3. Perform package setup (see "Setting up packages on the OS client" for a description of the procedure). Use the data from the planning worksheet to answer package setup prompts.

   After you complete package setup, **sysadm** prompts:

   ```
   Build kernel? [yes]
   ```

4. Answer **no**. (Otherwise, **sysadm** would build a custom kernel suitable for a diskless OS client.)

   ```
   Build kernel? [yes] no ↵
   Reboot now? [yes]
   ```

5. Do not reboot:

   ```
   Reboot now? [yes] no ↵

   local_root_swap
   DG/UX Operating System       Release n
   Console Login:
   ```

6. At the login prompt, enter the login name **sysadm**.

   ```
   local_root_swap
   DG/UX Operating System       Release n
   Console Login: sysadm ↵
   ```

   The screen displays a shell prompt (#) indicating a successful login.

7. Change the run level to **3**.

   ```
   # init 3 ↵
   ```

   A series of messages appears verifies transition to run level **3**.

8. Type the following command to link the new kernel to **/dgux** so that the correct kernel is rebooted each time you reboot your OS client.

```
# admkernel -o link local_root_swap ↵
```

Your OS client is now using a local / file system and swap space, and a remote **/usr** file system.

IMPORTANT: Because an OS client cannot load and set up a software package that resides in the **/usr** file system, those software packages must be located at the OS server.

Linking the OS client Internet address to the /tftpboot directory

You will delete the OS client name and then link its Internet address to the **tftpboot** directory. Perform these steps at the OS server:

1. From **sysadm**, follow this path to delete the OS client.

   ```
   Client-> OS Client-> Delete
   ```

2. Specify the OS client hostname (for example, **junior**) and press Enter.
3. Quit **sysadm**.
4. From the shell, link the OS client's Internet address to the **/tftpboot** directory using the following commands. Use the data from the planning worksheet for the OS client's Internet address in hexadecimal format. Be sure to use uppercase letters when specifying the Internet address. An example that shows the Internet address 80DE0356 follows.

   ```
   # cd /tftpboot ↵
   # ln -s /usr/stand/boot.aviion 80DE0356 ↵
   # initrarp ↵
   ```

   This link maintains OS client address entries that are resolved and used each time the OS client boots. The OS client (with a local / file system and swap space) fails to boot if this entry is not present in **/tftpboot** on the OS server.

   You have added an OS client that has a local root and swap virtual disk. To add another such OS client, repeat the steps in this section. If you want to add another type of client, go to the pertinent section in this chapter:

   - Adding a diskless OS client
   - Adding an OS client that has a local **swap** virtual disk

Adding an OS client that has a local swap virtual disk

This type of OS client has an attached disk drive that can provide its own local **swap** virtual disk OS resources while receiving its

root and **usr** virtual disk OS resources (/ and **/usr** file systems) from the OS server over a LAN.

These procedures assume the setup of a diskless OS client. You will modify the diskless OS client configuration (the entire OS resides remotely on the OS server's attached disk drive) so that the OS client continues to get its / and **/usr** file systems from the OS server, but its swap resources will be put on the OS client's attached disk drive.

IMPORTANT: If you have not already created a diskless OS client, create one following instructions in "Adding a diskless OS client" and then return to this section to continue.

Before you begin these procedures, make sure that:

- The AViiON computer to be used as an OS server is running the DG/UX system (its kernel has booted to a run level of 3) and is connected to a network.
- The computer system to be used as an OS client and its attached disk drive have been successfully powered up.
- The `SCM` prompt is displayed on the screen of the AViiON computer to be used as an OS client.
- You know the name of the disk drive attached to the OS client. The example used in this section is **sd(insc(0),3,0)**.

You will perform these steps:

- Build a temporary diskless OS client kernel.
- Boot the temporary kernel to a run level of 1.
- Build and boot an OS client kernel with local **swap** virtual disk resources.
- Set up packages and log in.

Building a temporary OS client kernel

Perform these procedures at the OS server:

1. Follow this path through **sysadm** to create a temporary system file.

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

Sysadm prompts

```
System configuration file name: [xxx]
```

2. Enter **temp**:

```
System configuration file name: [aviion] temp ↵
Build for this host or for OS client(s) of this
    host: [this host]
```

The name you type is appended to the supplied base filename **system** to yield the full name **system.temp**.

3. Enter **OS client**:

```
Build for this host or for OS client(s) of this
host: [this host] OS Client ↵
Boot Device: [inen(0)]
```

4. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0).** Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(***n***)**, where *n* is a number 0 through 5. For example:

```
Boot Device: [inen(0)]  dgen(0) ↵
Editor: [/usr/bin/vi]
```

5. Press Enter to accept the default editor, **/usr/bin/vi**:

```
Editor: [/usr/bin/vi] ↵
```

6. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section of the file follows, showing the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both. Determine which situation applies.

#       kbd()               # -- keyboard
#       grfx()              # -- graphics display
        sd(insc(),*)        # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)        # -- all SCSI tapes on integrated SCSI adapter
        inen()              # -- integrated Ethernet controller
#       duart()             # -- integrated Duart terminal line controller
#       duart(1)            # -- second Duart (if present on system)
#       lp()                # -- integrated printer controller (if present)
```

7. Save the file and quit the editor by typing **ZZ**. If asked about continuing, answer yes.

   Sysadm prompts:

   ```
   Link the new kernel for which clients:[all]
   ```

8. Enter the OS client's hostname, used when you created a diskless OS client. For example, **junior**:

   ```
   Link the new kernel for which clients:[all] junior ↵
   Save the old kernel? [yes]
   ```

9. If you can afford the disk space, we recommend you accept the **yes** default response.

   ```
   Save the old kernel? [yes] ↵
   Continue with the build? [yes]
   ```

10. To continue the build process, press Enter:

    ```
    Continue with the build? [yes] ↵
    Configuring system...
    Building kernel...
    Successfully built dgux.temp
    ```

    You have successfully built a temporary kernel named **dgux.temp**. Continue to the next section.

Booting the temporary kernel, creating an OS kernel with local swap resources, and setting up packages

Perform these procedures at the OS client.

1. Make sure that your system has been shut down. Your screen should display the SCM prompt. If it doesn't, shut down your system now using these commands:

```
# cd / ↵
# shutdown -g0 -y ↵
# halt ↵
SCM>
```

2. Boot the OS client's temporary kernel over the network to run level 1:

```
SCM> b inen(0) -1 ↵
```

Your screen displays a series of booting messages:

```
Booting inen( ) -1
Local Ethernet address is 08:00:1B:18:03
Local Internet address is 128.222.3.86 or 80DE0356 hex
...
```

A log-in prompt appears at the conclusion of the boot process. Log in as **sysadm.** The screen displays a shell prompt (#), indicating a successful login.

```
temp
DG/UX Operating System        Release n
Console Login: sysadm ↵
```

3. Set the TERM variable using the following Bourne shell commands:

```
# TERM=vt100 ↵
# export TERM ↵
```

4. Invoke **sysadm**:

```
# sysadm ↵
```

5. If the attached disk is new, make sure it is soft formatted and registered. For how to soft format and register a disk see *Managing Mass Storage Devices and DG/UX™ File Systems*. The soft format automatically registers the disk. Go to step 8.

6. If the disk has been used for swap storage, this means it uses logical disk format. You must convert it to virtual disk format. To do this, use the **sysadm** sequence `Device-> Disk-> Physical-> Convert`.

7. If the disk has been used for data storage, make sure the disk has 50,000 free blocks. To determine free disk space, use the **sysadm** sequence `Device-> Disk-> Physical-> List` and select the normal viewing style.

8. Create the **swap** virtual disk with a length of 50,000 blocks. You do not need to create a file system this virtual disk.
9. Quit **sysadm**.
10. Using **vi**, edit the **/etc/fstab** file. Insert a pound sign (#) in the first column of the line that begin with "server–hostname," to produce the following line:

    ```
    # server–hostname:/srv/swap/client–hostname  swap  nfs sw x 0
    ```

    The preceding line specifies a remotely mounted swap space from the OS server.
11. Next, insert the following line to specify a locally mounted swap area:

    /dev/dsk/swap swap_area swap sw 0 0
12. Type **ZZ** to save the file and quit the editor.
13. Follow this path through **sysadm** to create another system file:

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

    Sysadm prompts:

    ```
    System configuration file name: [xxx]
    ```
14. Enter **local_swap**:

    ```
    System configuration file name: [xxx] local_swap ↵
    Build for this host or for OS client(s) of this
        host: [this host]
    ```

    The name you type is appended to the supplied base name **system** to yield the full name **system.local_swap**.
15. Type **this host** and press Enter:

    ```
    Build for this host or for OS client(s) of this
    host: [this host] this host ↵
    Editor: [/usr/bin/vi]
    ```
16. Press Enter to accept the default editor, **vi**:

    ```
    Editor: [/usr/bin/vi] ↵
    ```
17. Search for the section in the configuration file labeled "OS Client Configuration Variables." Find the SWAPDEVTYPE variable and change its assigned value from NETWORK_SWAP to **LOCAL_SWAP**.

    ```
    OS Client Configuration Variables

    NETBOOTDEV          "inen()"
    ROOTFSTYPE          NETWORK_ROOT
    SWAPDEVTYPE         LOCAL_SWAP
    ```

18. Save the file and quit the editor by typing **ZZ**.

    After several process messages appear, **sysadm** prompts

    ```
    Link the new kernel to /dgux? [yes]
    ```

19. Link the new kernel to **/dgux** by answering yes:

    ```
    Link the new kernel to /dgux? [yes] ↵
    Continue with the build? [yes]
    ```

20. Continue the build process by pressing Enter:

    ```
    Continue with the build? [yes] ↵
    ```

    You have successfully built a kernel named **/dgux.local_swap** that has been linked to **/dgux**.

21. Using **sysadm** `System-> Kernel-> Reboot`, verify the boot path and boot the new kernel. Confirm your desire to reboot.

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

    The screen displays a series of booting messages during which you must confirm the current date and time. Booting leads automatically to the DG/UX package setup process.

    ```
    Booting inen( ) -i
    DG/UX System Release n Bootstrap
    Loading image ........
    .
    Linking short names for /dev device nodes .....
    Starting disk daemons ....
    Mounting local file systems .......
    Starting installation steps ....
    Set up packages [yes]
    ```

    IMPORTANT: Because it is located at the OS server, an OS client cannot load and set up a software package that resides in the **/usr** file system.

22. Respond to the **sysadm** package prompts as follows:

    ```
    Set up package(s)? [yes] ↵
    Package Name(s): [all] ↵
    OK to perform operation? [yes] ↵
    ```

    Perform package setup as explained in Chapter 13. Use the information from the OS client package setup worksheet that you used to set up the OS client as a diskless OS client.

    After you complete package setup, **sysadm** prompts:

```
Build kernel? [yes]
```

23. Answer **no**. (Otherwise, sysadm would build a custom kernel suitable for a diskless OS client.) For example:

```
Build kernel? [yes] no ↵
Reboot now? [yes]
```

24. Do not reboot:

```
Reboot now? [yes] no ↵

local_swap
DG/UX Operating System       Release n
Console Login:
```

25. At the login prompt, log in using the **sysadm** login name:

```
local_swap
DG/UX Operating System       Release n
Console Login: sysadm ↵
```

The screen displays a shell prompt (#), indicating a successful login.

26. Change the run level from **i** to **3**:

```
# init 3 ↵
```

A series of messages verifies transition to run level 3.

27. Type the following command to link the new kernel to **/dgux** so the correct kernel reboots each time you reboot your OS client.

```
# admkernel -o link local_swap ↵
```

Your OS client now uses a local swap space and a remote / and **/usr** file system.

You have added an OS client that has a local swap virtual disk. To add another such client, repeat the steps in this section. If you want to add another type of client, go to the pertinent section in this chapter:

- Adding a diskless OS client
- Adding an OS client that has local **root** and **swap** virtual disks

End of Chapter

19 X terminal clients

This chapter tells how to manage X terminal clients. An X terminal client is a type of graphics terminal that provides X Window System graphics support even though it does not have its own CPU. Lacking its own disk and operating system, the X terminal depends on an OS server for its secondary bootstrap.

Managing X terminal clients

In addition to OS clients, which are typically diskless workstations, the DG/UX system supports the AViiON AVX-30 network display station, referred to as an X terminal. An X terminal is a terminal with X Window System graphics capabilities. Unlike a normal user terminal, which connects to a computer's asynchronous communication port, the X terminal connects directly to the network.

Lacking its own disk drive and operating system, the X terminal client depends on an OS server for its bootstrap. The X terminal boots in a manner that is similar to an OS client in that it starts by announcing its need to boot over the network. The server assigned to the X terminal then transfers bootstrap code to the X terminal, allowing the X terminal to initialize itself and become an active member of the network. Unlike an OS client, an X terminal does not run an operating system that it receives from the server; instead, it runs its own network and graphics software sufficient to communicate with other hosts on the network and provide an X Window System graphics environment for the user.

The **sysadm** Client–> X Terminal menu provides operations for adding, deleting, modifying, and listing profiles for X terminals supported by your system. The Defaults menu is for managing different values for the default bootstrap that appears when you add an X terminal client.

To make an X terminal client operational, you perform these steps:

- Complete the planning worksheet
- Add an X terminal client to the **hosts** database
- Add an X terminal client to the **ethers** database
- Attach an X terminal client to its bootstrap file

Network planning for X terminal clients

An X terminal client network planning worksheet appears after this section to help you prepare for installation. It lists network

parameters for which you need to supply values. Completing the worksheet before you begin the installation procedures for clients will speed up the installation process.

If you have experience installing the DG/UX system or other operating systems, you may prefer to go directly to the X terminal client network planning worksheet.

The information necessary for completing the planning worksheet follows:

Host name	A unique name you assign to your X terminal hardware that can contain up to 31 alphabetic and numeric characters. However, you are advised to keep the names short. Host names that relate to the use or location of the system are particularly helpful in networked environments where hosts may share file systems. Examples of hostnames are **fred**, **jamaica**, and **writer_doc**. Do not use the capitalized names **MY_HOST** or **PRIMARY**; the system reserves these names.
Internet address	You provide the Internet address for the host being setup. An example of an Internet address is **128.223.2.1**. For information on Internet addresses, see *Managing TCP/IP on the DG/UX™ System*.
Ethernet address	Host address that is unique to the particular hardware. This address is preset at the factory. It consists of six 2-digit (hexadecimal) fields separated by colons in the form *nn:nn:nn:nn:nn:nn*. Refer to your hardware documentation for information on determining your Ethernet address.

X terminal client network planning worksheet

Figure 19–1 is a worksheet in which you record the Internet address and Ethernet address for each X terminal client on your system.

X Terminal Client Hostname	Internet Address	Ethernet Address
Example: xterm1	*Example: 123.227.3.15 (use periods)*	*Example: 08:00:1B:03:32:44 (use colons)*

Figure 19–1 X terminal client network planning worksheet

Adding an X terminal client

IMPORTANT: Before performing the Add operation, add the X terminal client's Internet and Ethernet addresses to your TCP/IP databases. The X terminal displays its Ethernet address when you turn on power. Get the Internet address from the person who manages the X terminal client. For how to add the Internet address, see "Adding an X terminal client to the OS server's Internet database. For how to add the Ethernet address, see "Adding an X terminal client to the OS server's Ethernet database."

Select the **sysadm** `Client-> X Terminal-> Add` operation to add support for an X terminal client. This operation sets your server up so that an AViiON AVX-30 network display station (or similar X terminal) can boot. This operation 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.

The Add operation prompts you for the host name by which the X terminal client is known on the network. The operation also prompts you for the pathname of the bootstrap file needed to boot the X terminal. The default bootstrap file is intended for AViiON AVX-30 network display stations. For other X terminals, you need to specify the pathname of a bootstrap file that is appropriate for that X terminal.

Adding an X terminal client to the OS server's Internet database

The **/etc/hosts** database contains a list of Internet address and X terminal client hostname pairs to allow network communication between the OS server and the X terminals.

1. To add an entry to the **hosts** database, follow this path through **sysadm** at the OS server:

   ```
   Networking-> TCP/IP-> Databases-> Hosts-> Add
   ```

   The **sysadm** prompt depends on whether your host is an OS server that has also been declared the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays:

   ```
   Hosts database to use: [NIS (YP)]
   ```

   If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Hosts database will be used.` **Sysadm** will let you insert a new entry in the *local* **hosts** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

   ```
   Hosts database to use: [NIS (YP)] ↵
   ```

3. **Sysadm** displays:

```
Host Name:
```

4. Consult your OS client network planning worksheet that you completed and enter the hostname; for example:

```
Host Name: xterm1 ↵
Internet Address:
```

5. From your OS client network planning worksheet, enter the four-field (four octet) internet address. For example:

```
Internet Address: 123.228.2.15 ↵
Alias List:
```

6. You may specify an alternate name (alias) for your host. For example, if your department uses long hostnames, you may choose a shorter alias for convenience. For example:

```
Alias List: x1 ↵
OK to perform operation?  [yes]
```

7. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example:

```
OK to perform operation?  [yes] ↵
xterm1 has been added.
```

Sysadm has added the X terminal to the appropriate hosts file. Repeat the steps above for each X terminal client you want to add to the **hosts** database.

Adding an X terminal client to the OS server's Ethernet database

The **/etc/ethers** database contains a list of Ethernet address and OS client hostname pairs to allow network communication between the OS server and its X terminal clients. To add an entry to the **ethers** database, perform these steps at the OS server.

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

```
Networking-> TCP/IP-> Databases-> Ethers-> Add
```

The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays:

```
Ethers database to use: [NIS (YP)]
```

If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master.Only the local Ethers database will be used.` **Sysadm** will let you insert a new entry in the *local* **ethers** database for each OS client. Skip to step 2.

2. For a networked environment, press Enter to accept the default value **NIS**:

```
Hosts database to use: [NIS (YP)] ↵
```

3. **Sysadm** displays:

```
Host Name:
```

4. Consult the OS client network planning worksheet that you completed and enter the hostname; for example:

```
Host Name: xterm1 ↵
Ethernet Address:
```

5. From your OS client network planning worksheet, enter the Ethernet address, form *hh:hh:hh:hh:hh* where *h* is a hexadecimal number. For example:

```
Ethernet Address: 08:00:1B:03:32:44 ↵
OK to perform operation?  [yes]
```

6. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example:

```
OK to perform operation?  [yes] ↵
xterm1 has been added.
```

Sysadm has added the X terminal to the appropriate **ethers** file. Repeat this procedure to add as many X terminals as desired to the **ethers** database.

Attaching the X terminal client to its bootstrap file

To attach an X terminal client to its bootstrap file, follow these steps at the OS server.

1. Follow this path through **sysadm** at the OS server.

```
Client-> X Terminal-> Add
```

Sysadm prompts:

```
Client Host Name:
```

2. Specify the hostname of the X terminal client. For example:

```
Client Host Name: xterm1 ↵
Bootstrap File: [/usr/opt/X11/xtd/avx30boot]
```

3. **Sysadm** will create a link to the bootstrap file for this X terminal client. Press Enter to accept the default bootstrap file pathname. For example:

```
Bootstrap File: [/usr/opt/X11/xtd/avx30boot] ↵
OK to perform operation? [yes]
```

4. If you are happy with the answers, confirm with Enter:

```
OK to perform operation? [yes] ↵
Xterminal xterm1 has been added.
```

You have added an X terminal client and attached it to the bootstrap file. The following actions take place when you turn on your X terminal client.

- The X terminal issues an **rarp** request on the network. The host you have set up matches the **ethers** address that is part of the **rarp** request to the X terminal hostname in the **ethers** database.
- The host matches the X terminal hostname to the Internet address in the **hosts** database. The host, in response to the **rarp** request, returns the Internet address to the X terminal client.
- Finally, the X terminal client uses its Internet address to access the correct file in the **/tftpboot** and downloads its Xserver.

After the X terminal client's Xserver loads, you are able to access the X terminal's configuration menus. For more information, refer to the AVX-30 Release Notice in the **/usr/opt/X11/release** directory.

For information on adding a user account for the X terminal client user and assigning a password, refer to Chapter 5.

Deleting an X terminal client

Select the **sysadm** operation `Client-> X Terminal-> Delete` to delete the files on your server that allow an X terminal client (such as the Data General AVX-30 network display station) to boot.

Select **all** to delete all X terminal clients, or specify the name of a single X terminal client.

After deleting an X terminal client, you may wish to remove the client's Internet and Ethernet addresses from your TCP/IP databases. See the Networking menu.

Modifying X terminal clients

Select the **sysadm** operation `Client-> X Terminal-> Modify` to modify the Modify operation to change the bootstrap file pathname for X terminal clients supported by your system. To change the bootstrap file for all clients, select **all**. To change the bootstrap file for just one X terminal client, select the client's host name.

Listing X terminal clients

Select the **sysadm** operation `Client-> X Terminal-> List` to display parameters associated with the X terminal clients supported by your server. The fields in the display are:

`Client Name`
: The host name by which the X terminal client is known on the network.

`Address`
: The client's Internet address.

`Bootstrap`
: The client's Internet address, in hexadecimal, which is used as a link file pointing to the bootstrap file.

`Linked to`
: The pathname of the bootstrap file to which the link points.

Setting X terminal client defaults

The `Client-> X Terminal-> Defaults` menu provides operations for creating, removing, modifying, and listing default sets. A default set is a value (for the bootstrap file pathname) that can appear as the default when you add an X terminal client. An additional operation allows you to select which set determines the default currently used by the Add operation.

Creating a default set

Select the `Create` operation to make a new default set. Use the Modify operation to define the bootstrap value in the default set.

Creating an X terminal client default set does not automatically make the set the current one; you need to use the Select operation to make it current.

Removing a default set

Select the `Remove` operation to delete an existing default set. Deleting a default set has no effect on X terminal clients added when the default set was in use.

Modifying a default set

Select the `Modify` operation to set or change the value of the bootstrap parameter in an existing default set. Modifying a default set has no effect on X terminal clients added when the default set was in use.

Listing default sets

Select the `List` operation to display the bootstrap parameter setting in an existing default set.

Selecting a default set

Use the `Select` operation to set the current X terminal clients default set to be used by the Add operation. Creating an X terminal client default set does not automatically make that set the current one; you need to use the `Select` operation to make it current.

End of Chapter

20 DG/UX device names

Each DG/UX system device has a unique name which is constructed using the DG/UX device naming format. This chapter discusses standard devices and explains device names for the following devices:

- Network controller device names
- Synchronous line controller names
- Asynchronous line controller names
- Device names for keyboards, graphics displays, and parallel line printers
- Device nodes for terminals

For explanations of the following device naming formats, see *Managing Mass Storage Devices and DG/UX™ File Systems*:

- SCSI disk and tape device names
- SMD and ESDI disk drive names
- Non-volatile error correcting memory device names
- Device nodes for physical, virtual, and logical disks, and for tape drives

Short and long device names

Two forms of the DG/UX device-naming format are available for specifying devices: short and long.

You usually use the short form to specify devices. However, each short form is internally represented by its long form. Syntax for both forms is given in device-specific sections in this chapter.

Network controller device names

The format for the network controller devices is:

device [@*device-code*] ([*vme-controller*] [*controller-address* [,*secondary-address* [,*alternate-address*]]])

where:

device is a mnemonic that specifies the network controller device. The valid values are:

cien CMC VME Ethernet intelligent controller

dgen Data General second generation integrated Ethernet controller

hken Interphase VME Ethernet controller

inen Integrated Ethernet controller

pefn Interphase VME Fiber Distributed Data Interface (FDDI) controller

vitr VME token ring controller

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it is the interrupt vector.

vme-controller specifies the VME controller channel, for example, **(vme(1))**; needed only if the computer has more than one VME channel and this device is on a channel other than the first one. If you omit this, **vme(0),** is implied. For example, the device name **pefn(vme(0),1)** is functionally the same as **pefn(1)**. This item does not apply to the integrated controllers **dgen** and **inen**.

controller-address Valid controller addresses are:

cien Controller's VME A32 address

dgen Not used

hken Interphase VME Ethernet controller

inen Not used

pefn Controller's VME A16 address

vitr Controller's VME A32 address

Table 20–1 lists the valid controller numbers supported (short form), the device code, and controller address (long form).

secondary-address is different for each device; they follow:

cien Not used

dgen Not used

hken Controller's VME A32 address

inen Not used

pefn Not used

vitr Alternate token ring address that overrides board-set default

Table 20–1 lists the valid controller numbers supported (short form), device code, and controller address (long form). Controllers whose values fall outside this range are considered nonstandard.

alternate-address is different for each device; they are:

cien Alternate Ethernet address that overrides board-set default

dgen Alternate Ethernet address that overrides board-set default

hken Alternate Ethernet address that overrides board-set default

inen Alternate Ethernet address that overrides board-set default

pefn Not used

vitr IBM product ID (18 hex digits) that overrides board-set default

Table 20–1 Network controller names

Controller Number (Short Form)	Controller Address (Long Form)
CMC Ethernet Controller (cien)	
cien(0)	cien@48(E1800000)
cien(1)	cien@49(E1900000)
cien(2)	cien@4A(E1A00000)
cien(3)	cien@4B(E1B00000)
cien(4)	cien@4C(E1C00000)
cien(5)	cien@4D(E1D00000)
cien(6)	cien@4E(E1E00000)
cien(7)	cien@4F(E1F00000)
Data General Second Generation Integrated Ethernet Controller (dgen)	
dgen(0)	Not used.
dgen(1)	Not used.
dgen(2)	Not used.
dgen(3)	Not used.
dgen(4)	Not used.
dgen(5)	Not used.
Interphase VME Ethernet Controller (hken)	
hken(0)*	hken@15(FFFF4000,E1000000)
hken(1)*	hken@16(FFFF5000,E1080000)
hken(2)	hken@10(FFFF4200,E1100000)
hken(3)	hken@11(FFFF4400,E1180000)
hken(4)	hken@12(FFFF4600,E1200000)
hken(5)	hken@13(FFFF4800,E1280000)
hken(6)	hken@14(FFFF4A00,E1300000)
hken(7)	hken@17(FFFF4C00,E1380000)
* For computers limited to 1 Gbyte or less of physical memory, these are hken(0) hken@15(FFFF4000,55900000) hken(1) hken@16(FFFF5000,55980000)	
Integrated Ethernet Controller (inen)	
inen(0)	Not used.
Interphase VME FDDI Controller (pefn)	
pefn(0)	pefn@78(FFFF2000)

Continued

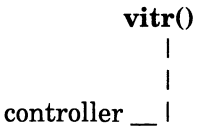
Table 20–1 Network controller names

Controller Number (Short Form)	Controller Address (Long Form)
Interphase VME FDDI Controller (pefn)	
pefn(0)	pefn@78(FFFF2000)
pefn(1)	pefn@79(FFFF2200)
pefn(2)	pefn@7A(FFFF2400)
pefn(3)	pefn@7B(FFFF2600)
pefn(4)	pefn@7C(FFFF2800)
pefn(5)	pefn@7D(FFFF2A00)
pefn(6)	pefn@7E(FFFF2C00)
pefn(7)	pefn@7F(FFFF2E00)
VME Token Ring Controller (vitr)	
vitr(0)*	vitr@40(E4000000)
vitr(1)*	vitr@41(E4002000)
vitr(2)	vitr@42(E4004000)
vitr(3)	vitr@43(E4006000)
vitr(4)	vitr@44(E4008000)
vitr(5)	vitr@45(E400A000)
vitr(6)	vitr@46(E400C000)
vitr(7)	vitr@47(E400E000)
* For computers limited to 1 Gbyte or less of physical memory, these are vitr(0) vitr@40(61000000) vitr(1) vitr@41(61002000)	

Controllers whose values fall outside this range are considered nonstandard.

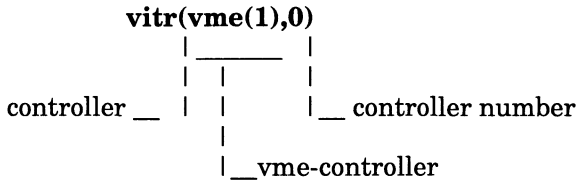
Examples 1 and 3 are short and long forms of the same device.

Example 1: VME token ring controller (short form)



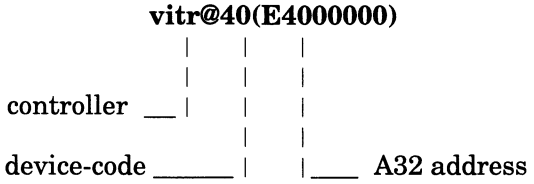
This device name identifies the first VME token ring controller.

Example 2: VME token ring controller on second vme controller channel (short form)



This device name identifies the first the first VME token ring controller on the second VME channel.

Example 3: VME token ring controller (long form)



This device name identifies a VME token ring controller whose device code is 40 and A32-address is E4000000.

Other examples

hken()	First () Interphase VME Ethernet LAN.
pefn(1)	Second (1) Interphase FDDI controller.

Synchronous line controller names

The format for the synchronous line controllers is:

device (*[vme–controller] controller-number*)
or
device@device–code (*[vme–controller] VME A32 address*)

where:

device is a mnemonic that specifies the synchronous line controller. The valid values are:

iscd	Integrated synchronous chip controller
ssid	VME VSC synchronous controller
vsxb	VME VSC/3i synchronous controller

vme-controller specifies the VME controller channel, for example, **vme(1)**; needed only if the computer

has more than one VME channel and this device is on a channel other than the first one. If you omit this, **vme(0),** is implied. For example, the device name **vsxb(vme(0),1)** is functionally the same as **vsxb(1)**. This does not apply to the integrated controller **iscd**.

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it represents the interrupt vector.

controller-number identifies the controller the device is attached to.

VME A32 address identifies the address of the controller that the device is attached to.

Table 20–2 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form).

Asynchronous line controller names

The format for asynchronous line controllers is:

device[@device-code]([vme–controller] controller-number, line-number)

where:

device is a mnemonic that specifies the asynchronous line controller. The valid values are:

duart Dual-line asynchronous terminal controller

syac Systech VME asynchronous terminal controller

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it represents the interrupt vector.

vme-controller specifies the VME controller channel, for example, **(vme(1)**; needed only if the computer has more than one VME channel and this device is on a channel other than the first one. If you omit this, **vme(0),** is implied. For example, the device name **syac(vme(0),1)** is functionally the same as **syac(1)**. This does not apply to the integrated controller **duart**.

controller-number identifies the controller to which the device is attached.

VME A32 address identifies the address of the controller that the device is attached to.

IMPORTANT: For the **duart** device, it refers to the address of its memory-mapped control registers (when using the long form). See Table 20–2.

line-number identifies the particular line attached to the controller, starting at 1. For **duart**, either 0 or 1.

Table 20–2 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form). Controllers whose values fall outside this range are considered nonstandard.

Table 20–2 Synchronous and asynchronous line controller device names

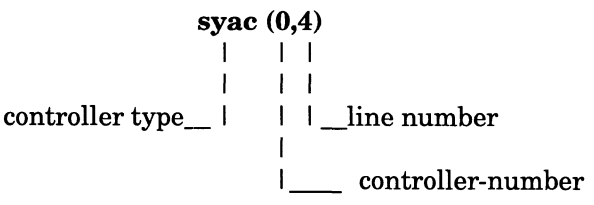
Controller Number (Short Form)	Controller Address (Long Form)
Integrated Synchronous Chip Controllers	
iscd()	Not used.
VME VSC Synchronous Controller (ssid)	
ssid(0)	ssid@50(55B00000)
ssid(1)	ssid@51(55B10000)
ssid(2)	ssid@52(55B20000)
ssid(3)	ssid@53(55B30000)
ssid(4)	ssid@54(55B40000)
ssid(5)	ssid@55(55B50000)
ssid(6)	ssid@56(55B60000)
ssid(7)	ssid@57(55B70000)
ssid(8)	ssid@58(E2080000)
ssid(9)	ssid@59(E2090000)
ssid(A)	ssid@5A(E20A0000)
ssid(B)	ssid@5B(E20B0000)
ssid(C)	ssid@5C(E20C0000)
ssid(D)	ssid@5D(E20D0000)
ssid(E)	ssid@5E(E20E0000)
ssid(F)	ssid@5F(E20F0000)
VME VSC/3i Synchronous Controller (vsxb)	
vsxb(0)	vsxb@90(E3400000)
vsxb(1)	vsxb@91(E3410000)
vsxb(2)	vsxb@92(E3420000)
vsxb(3)	vsxb@93(E3430000)
vsxb(4)	vsxb@94(E3440000)
vsxb(5)	vsxb@95(E3450000)
vsxb(6)	vsxb@96(E3460000)
vsxb(7)	vsxb@97(E3470000)
vsxb(8)	vsxb@98(E3480000)
vsxb(9)	vsxb@99(E3490000)
vsxb(A)	vsxb@9A(E34A0000)
vsxb(B)	vsxb@9B(E34B0000)
vsxb(C)	vsxb@9C(E34C0000)
vsxb(D)	vsxb@9D(E34D0000)
vsxb(E)	vsxb@9E(E34E0000)
vsxb(F)	vsxb@9F(E34F0000)

Continued

Controller Number (Short Form)	Controller Address (Long Form)
Systech VME Asynchronous Terminal Controller (syac)	
syac(0)*	syac@60(E3000000)
syac(1)*	syac@61(E3020000)
syac(2)*	syac@62(E3040000)
syac(3)*	syac@63(E3060000)
syac(4)*	syac@64(E3080000)
syac(5)	syac@65(E30A0000)
syac(6)	syac@66(E30C0000)
syac(7)	syac@67(E30E0000)
syac(8)	syac@68(E3100000)
syac(9)	syac@69(E3120000)
syac(A)	syac@6A(E3140000)
syac(B)	syac@6B(E3160000)
syac(C)	syac@6C(E3180000)
syac(D)	syac@6D(E31A0000)
syac(E)	syac@6E(E31C0000)
syac(F)	syac@6F(E31E0000)
* For computers limited to 1 Gbyte or less of physical memory, these are	
syac(0)	syac@60(60000000)
syac(1)	syac@61(60020000)
syac(2)	syac@60(60040000)
syac(3)	syac@61(60060000)
syac(4)	syac@61(60080000)
Dual–Line Asynchronous Terminal Controller (duart)	
duart(0)	duart@4(FFF82000)
duart(1)** ***	duart@10(FFF82040)
duart(2)** ***	duart@17(FF782000)
duart(3)** ***	duart@18(FF782040)
** Not all computers support this duart. *** For AViiON 300-series and 400-series systems, duart(1) is duart@10(FFF82C00). For AViiON 530 systems, duart(1) is duart@10(FFF83220).	

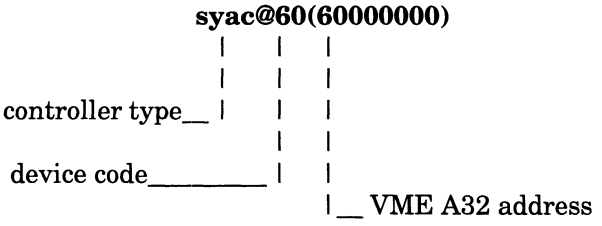
Examples 1 and 2 are short and long forms of the same device.

Example 1: Asynchronous terminal controller (short form)



This device name identifies the first (0) **syac** controller on the fourth (4) line.

Example 2: Asynchronous terminal controller (long form)

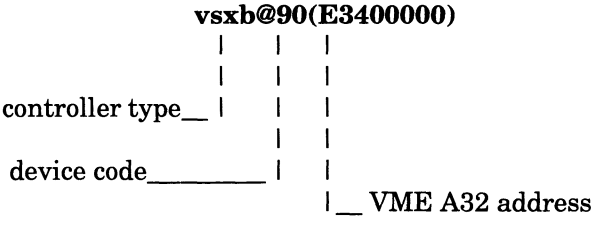


This device name identifies the **syac** controller whose device code is 60 and whose VME A32 address is 60000000.

Example 3: VME VSC synchronous interface controller (long form)

This device name identifies the **ssid** controller whose device code is 50, and VME-A32 address is 55B00000.

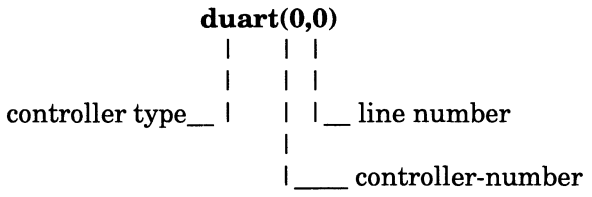
Example 4: VME VSC/3i synchronous controller (long form)



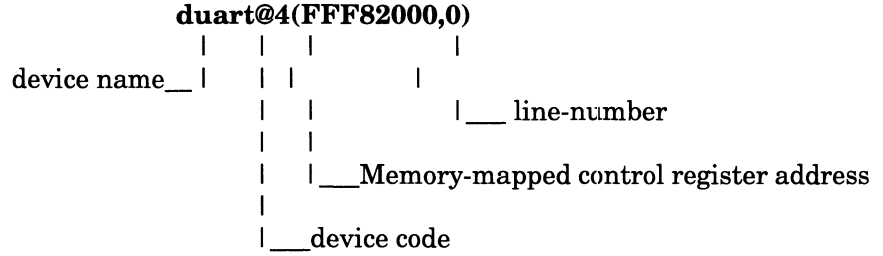
This device name identifies the **vsxb** controller whose device code is 90, and VME-A32 address is E3400000.

Examples 5 and 6 are short and long forms of the same device.

Example 5: Duart (short form)



Example 6: Duart (long form)



This device name identifies the first (0) device attached to the **duart** whose device code is 4 and memory-mapped control register address is FFF82000.

This device name identifies the first (0) device attached to the first (0) **duart**. This device could also be specified as **duart( )**.

Device names for keyboards, graphics displays, and printers

The format for other I/O devices is:

device [*@device-code*]

where:n

device is a mnemonic that specifies the particular device. The valid values are:

kbd	keyboard
grfx	graphics display
lp	parallel line printer

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it represents the interrupt vector.

Table 20–3 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form). I/O devices with values outside this range are nonstandard.

Table 20–3 Device names for keyboards, graphics displays, and printers

Controller Name (Short Form)	Controller Address (Long Form)
Graphic Display, Keyboard, and Line Printer (grfx, kbd,lp)	
grfx(0)	Not used
kbd(0)	Not used
lp(0)	Not used

Device nodes

This section describes device nodes for terminals. For information on device nodes for physical, virtual, and logical disks, and for tape drives, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

A device node is a special file that provides a handle to a particular device so that programs can access it. Device nodes are automatically created in the **/dev** directory each time you boot the kernel.

Device nodes come in two forms:

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

See the **mknod**(1M) manual page.

Terminal nodes

Terminal line devices have nodes with names of the form **/dev/tty***number*, where *number* is a decimal number of at least two digits starting at 00. The assignment of names to your system's terminal lines depends on the order that the terminal line controller devices are configured by the kernel, which in turn depends on the order those controller devices are listed in the system file. Terminal nodes are assigned in a first-come, first-served order, starting at **/dev/tty00** and working up. If you use the standard kernel auto configuration mechanism, all standard **duart** controllers on the system are listed in order, followed by all standard **syac** controllers on the system.

Only one asynchronous line is available on some **duart** devices. On stand-alone computer systems, this is because one channel of the first **duart** is used to provide access to the mouse device, which has the special device node **/dev/mouse**. Other systems also reserve one

duart channel for special use, in this case as the system console (**/dev/syscon**).

A **syac** controller causes from 16 to 255 device nodes to be created, depending on how many lines are on the controller. If a cluster controller box is used, all 255 lines are created, even though only a few of them (perhaps 8 or 16) are actually in use.

End of Chapter

Glossary

backup cycle list
A plan for doing daily, weekly, and monthly backups on tape. A default backup cycle list is supplied with the DG/UX system.

directory structure
Arrangement of hierarchically structured file systems. The DG/UX directory structure base is the root directory, pathname /. Directories within the root include **usr**, **bin**, and **opt**, with pathnames **/usr**, **/bin**, and **/opt**.

file system
A software-formatted portion of disk space typically located on a virtual disk. The file system contains the internal data structures that the operating system requires to keep track of files and directories on the virtual disk. For information on creating and managing file systems, see *Managing Mass Storage Devices and DG/UX™ File Systems*.

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**. System ID numbers are 1 to 99.

home directory
The origination 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 name.

init
A program that starts system processes based on entries 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.

initial program
The program invoked at the time the user logs in that provides a command line interface to the operating system. Choices are the Bourne shell (**/sbin/sh**), the C shell (**/sbin/csh**), the Korn shell (**/usr/bin/ksh**), and the restricted shell (**/bin/restsh**).

login name
A valid name for logging on to the system. Also known as a *user name*. A login name can be up to 32 characters, must

begin with an alphabetic character, and can contain alphanumeric characters (**a** through **z**, **A** through **Z**, **0** through **9**), dashes (-), underscores (_), and periods (.).

mail alias
A mailing list containing one or more login names or aliases. Mail addressed to a mail alias is delivered to all users listed in that mail alias.

mount point
The placement of a file system within the DG/UX directory structure. Mounting a file system attaches it to a directory and makes it accessible to users. At startup, the DG/UX system automatically mounts all file systems specified in the file system table file, **/etc/fstab**.

NIS master server
The NIS master server is the single system in the network that holds the master networks and hosts files that are exported to other systems. Global changes can be made only on this master system.

password
A unique string of alphabetic or numeric characters that allows a user access to the system. A password must be between six and eight characters long. At least one character must be a numeral or a special character.

password aging
An option that forces users to change their passwords within a specified number of weeks. When this time period lapses, the password no longer allows entry to the system and the user must choose a new password.

primary release
The release of the DG/UX system that runs routinely on a computer system, or that is accessed by most users of the computer system. See also *secondary release*.

rc scripts
The run command (rc) scripts that are executed at every boot and every time you change run levels. These scripts kill or start system services.

registration
The identification of a physical disk to the DG/UX operating system with the **sysadm** utility. You can register the disk when you software format it. At startup, the operating system automatically registers each disk built into its kernel, unless the disk is already registered by another host.

run level
A group of services provided by the system. Also called *run*

states or *run modes*. A run level can include services such as network-related capabilities, accounting, **cron** batch job scheduling, and line printer services. Typically, run level S (for single-user mode) provides no services, and run levels 1 through 3 provide increasing levels of system functionality. By default, DG/UX provides full multi-user and network capabilities at run level 3.

SAF The Service Access Facility, a set of features that let you manage ports, setting terminal type, mode, speed, and line discipline characteristics for the port. SAF can also start service programs for ports. A major function of SAF is to control user terminal lines, starting the **login** service for users who need to log in.

SCM The System Control Monitor, a command line interface run when no operating system is running on your computer (it is built into the hardware). From the SCM prompt (**SCM>**), you use the **b** command to boot the DG/UX kernel program, thereby starting the DG/UX operating system. For information on SCM, type **h** at the SCM prompt or see *Testing and Troubleshooting AViiON® Computers: AV/Alert and the AViiON® System Control Monitor.*

secondary release
An alternate operating system installed in an area accessible to users and client systems that need to run a different release of the DG/UX operating system or a different operating system.

sysadm
A utility program supplied with the DG/UX operating system for performing system management tasks. **Sysadm** runs as either a menu-driven or window-based program. There is a version you can run from DG/UX and a stand-alone version (path **/usr/stand/sysadm**).

user group
A set of users with common access privileges to a common set of files based on group ID numbers. Also known as a *group name*. User names for people that need access to the same files can be listed in a group. For example, anyone in group **prog** can access files associated with that group name, assuming the group permissions were set correctly for the files.

user ID

A unique number that identifies a user to the system. A valid user ID number (**uid**) is between 100 and 60000 and must not include a comma. The default superuser user ID (**sysadm** and **root**) is 0. System ID numbers are 1 to 99.

End of Glossary

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a. MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   b. Send your order form with payment to:

   Data General Corporation
   ATTN: Educational Services/TIPS G155
   4400 Computer Drive
   Westboro, MA 01581–9973

   c. TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   b. Check or Money Order – Make payable to Data General Corporation. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
Over 200 Items	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$0–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

Managing the
DG/UX™ System

093–701088–05

Cut here and insert in binder spine pocket