DataGeneral

Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Managing the DG/UX™ System

093–701088–02

AViiON®

PRODUCT LINE

Managing the DG/UX™ System

093–701088–02

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

Copyright ©Data General Corporation, 1991, 1992
All Rights Reserved
Unpublished - all rights reserved under the copyright laws of the United States.
Printed in the United States of America
Rev. 02, *August*, 1992
Licensed Material – Property of copyright holder(s)
Ordering No. 093–701088–02

Notice

DATA GENERAL CORPORATION (DGC) HAS PREPARED AND/OR HAS DISTRIBUTED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF THE COPYRIGHT HOLDER(S); AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE APPLICABLE LICENSE AGREEMENT.

The copyright holder(s) reserve the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS, AND THE TERMS AND CONDITIONS GOVERNING THE LICENSING OF THIRD PARTY SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE APPLICABLE LICENSE AGREEMENT. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

All software is made available solely pursuant to the terms and conditions of the applicable license agreement which governs its use.

Restricted Rights Legend: Use, duplication, or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at [DFARS] 252.227-7013 (October 1988).

Data General Corporation
4400 Computer Drive
Westboro, MA 01580

AViiON, CEO, DASHER, DATAPREP, DESKTOP GENERATION, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, GENAP, INFOS, microNOVA, NOVA, PRESENT, PROXI, SWAT, TRENDVIEW, and **WALKABOUT** are U.S. registered trademarks of Data General Corporation; and **AOSMAGIC, AOS/VSMAGIC, AROSE/PC, ArrayPlus, AV Object Office, AV Office, BaseLink, BusiGEN, BusiPEN, BusiTEXT, CEO Connection, CEO Connection/LAN, CEO Drawing Board, CEO DXA, CEO Light, CEO MAILI, CEO Object Office, CEO PXA, CEO Wordview, CEOwrite, COBOL/SMART, COMPUCALC, CSMAGIC, DASHER/One, DASHER/286, DASHER/286-12c, DASHER/286-12j, DASHER/386, DASHER/386-16c, DASHER/386-25, DASHER/386-25k, DASHER/386SX, DASHER/386SX-16, DASHER/386SX-20, DASHER/486-25, DASHER/LN, DATA GENERAL/One, DESKTOP/UX, DG/500, DG/AROSE, DGConnect, DG/DBUS, DG/Fontstyles, DG/GATE, DG/GEO, DG/HEO, DG/L, DG/LIBRARY, DG/UX, DG/XAP, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/3500, ECLIPSE MV/5000, ECLIPSE MV/5500, ECLIPSE MV/5600, ECLIPSE MV/7800, ECLIPSE MV/9300, ECLIPSE MV/9500, ECLIPSE MV/9600, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/30000, ECLIPSE MV/40000, FORMA-TEXT, GATEKEEPER, GDC/1000, GDC/2400, Intellibook, microECLIPSE, microMV, MV/UX, PC Liaison, RASS, REV-UP, SLATE, SPARE MAIL, SUPPORT MANAGER, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE,** and **XODIAC** are trademarks of Data General Corporation.

UNIX is a U.S. registered trademark of Unix System Laboratories, Inc. **NFS** is a U.S. registered trademark and **ONC** is a trademark of Sun Microsystems, Inc. The Network Information Service (NIS) was formerly known as Sun Yellow Pages. The functionality of the two remains the same; only the name has changed. The name **Yellow Pages** is a registered trademark in the United Kingdom of British Telecommunications plc and may not be used without permission. Portions of Chapter 9 and Chapter 11 are copyrighted by American Telephone and Telegraph Company. MS-DOS is a U.S. registered trademark of Microsoft Corporation.

Data General expects to release the VLC/VME Intelligent Internet Controller (VLCi), the VSC/3i Controller, VME FDDI Controller, and the Model 6760 and 6761 Cartridge Tape Drive in October 1992.

Managing the DG/UX™ System
093-701088-02

Revision History:		Effective with:
Original Release	– June, 1991	DG/UX, Release 5.4
Revision 1	– January, 1992	DG/UX, Release 5.4.1
Revision 2	– August, 1992	DG/UX, Release 5.4.2

A vertical bar in the margin of a page indicates substantive technical change from the previous revision. (The exception is Appendix D, which contains entirely new material.)

Preface

This manual introduces DG/UX™ system management and provides complete coverage of the various DG/UX system-related tasks that the typical system administrator must address. The manual does not cover installation. To install your DG/UX operating system software, see *Installing the DG/UX™ System*. For a streamlined treatment of the tasks you perform to make a freshly-installed system usable, see *Customizing the DG/UX™ System.*

The DG/UX system provides the **sysadm**(1M) utility to help you with installation and management tasks. The **sysadm** utility is a menu-based interface to a number of programs intended to simplify the responsibilities of a system administrator. Using **sysadm** and its attendant programs, you can do things such as load and set up software packages, add user profiles, manage the network and printers, and build customized DG/UX kernels. This manual tells how you can use **sysadm** to help with these and other tasks.

Manual Outline

This manual contains the following chapters and appendixes:

Chapter 1	Introduction to system administration. Looking at the DG/UX system feature-by-feature, provides a brief tour through the various areas of system management.
Chapter 2	Overview of routine management tasks. Covers tasks that you should perform every time you boot the system and then periodically while the system is running.
Chapter 3	Operating the DG/UX system, startup and shutdown, recovering from system trouble (such as crashes and power outages). Explains run command scripts and run levels.
Chapter 4	Monitoring system activity, setting system parameters and time and date, configuring the system, and rebuilding the kernel on a routine basis.
Chapter 5	Managing software, creating releases, loading and setting up software packages.
Chapter 6	Managing OS and X terminal clients, setting defaults, adding and deleting OS and X terminal clients.
Chapter 7	Managing disks using **sysadm** and the disk management utility, **diskman**. Format physical disks, create logical disks and file systems, check file systems, change file system size, display disk information, update the operating system, and do various other disk-related tasks. The chapter also covers special disk- and file system-related features such as software-mirrored disks, cached disks, failover disks, data striping, fast recovery file systems, and write verification.
Chapter 8	DG/UX and MS-DOS® file system management. Shows how to mount, unmount, add, and delete file systems, manage swap areas, list information

	about file systems, search for possible security problems, make backup tapes, and check the consistency of file systems.
Chapter 9	Setting up and managing terminals and ports.
Chapter 10	Managing your LAN, Sync, and VDA controllers.
Chapter 11	Managing printers. Contains operations for managing printing devices, classes, filters, forms, requests, and the scheduler.
Chapter 12	Network terms and definitions; basic network management functions: setting parameters for the ONC™/NFS®, TCP/IP, and UUCP systems. Also covers setting up UUCP files and directories.
Chapter 13	Adding login accounts, creating aliases and groups. Explains how these are used and managed differently in stand-alone and OS server-client environments.
Chapter 14	Monitoring the use of system resources using the accounting facility; printing various kinds of summary reports.
Chapter 15	Using CD-ROM, magneto-optical, and SCSI diskette drives.
Appendix A	UUCP error and status messages.
Appendix B	File system checker (**fsck**(1M)) error conditions.
Appendix C	Reference tables summarizing commands used with SAF, discussed in Chapter 9.
Appendix D	Selective summary of directories and files of interest to the system administrator.

Readers, Please Note

Data General manuals use certain symbols and styles of type to indicate different meanings. The Data General symbol and typeface conventions used in this manual are defined in the following list. You should familiarize yourself with these conventions before reading the manual.

This manual also presumes the following meanings for the terms "command line," "format line," and "syntax line." A command line is an example of a command string that you should type verbatim; it is preceded by a system prompt and is followed by a delimiter such as the curved arrow symbol for the New Line key. A format line shows how to structure a command; it shows the variables that must be supplied and the available options. A syntax line is a fragment of program code that shows how to use a particular routine; some syntax lines contain variables.

Convention	Meaning
boldface	In command lines and format lines: Indicates text (including punctuation) that you type verbatim from your keyboard. All DG/UX commands, pathnames, and names of files, directories, and manual pages also use this typeface.
`constant width/ monospace`	Represents a system response on your screen. Syntax lines also use this font.

Licensed Material – Property of copyright holder(s)

italic	In format lines: Represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands. In text: Indicates a term that is defined in the manual's glossary.
[ ]	In format lines: These brackets surround an optional argument. Don't type the brackets; they only set off what is optional. The brackets are in regular type and should not be confused with the boldface brackets shown below.
[ ]	In format lines: Indicates literal brackets that you should type. These brackets are in boldface type and should not be confused with the regular type brackets shown above.
...	In format lines and syntax lines: Means you can repeat the preceding argument as many times as desired.
$ and %	In command lines and other examples: Represent the system command prompt symbols used for the Bourne and C shells, respectively. Note that your system might use different symbols for the command prompts.
↵	In command lines and other examples: Represents the Enter key, which is the name of the key used to generate a new line. (Note that on some keyboards this key might be called New Line or Return instead of Enter.) Throughout this manual, a space precedes the Enter symbol to improve readability; you can ignore it.
< >	In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as <**Ctrl–D**>, <**Esc**>, and <**3dw**>) from surrounding text. Note that these angle brackets are in regular type and that you do not type them; there are, however, boldface versions of these symbols (described below) that you do type.
<, >, >>	In text,command lines, and other examples: These boldface symbols are redirection operators, used for redirecting input and output. When they appear in boldface type, they are literal characters that you should type.
□	In command lines and other examples: Represents the cursor, which indicates your current typing position on the screen.

Contacting Data General

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

Manuals

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

Licensed Material – Property of copyright holder(s)

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

Licensed Material – Property of copyright holder(s)

Contents

Licensed Material – Property of copyright holder(s)

Tables

Figures

Figure

Chapter 1
Introduction to System Management

This chapter introduces administrative responsibilities in terms of the features of the DG/UX system. The purpose is not to teach you procedures at this early stage, but rather to give you an idea of the areas that you can investigate as you arrive at your own grasp of your responsibilities. All of the tasks discussed in this chapter may not apply to your system.

Your primary tool for performing system administration tasks is the **sysadm**(1M) utility. The **sysadm** utility provides a number of operations, organized in a menu hierarchy, that you use to set up printers, create user profiles, allocate disk space, perform backups, set network parameters, and so on. You can invoke **sysadm** to provide a scrolling character interface for normal terminals or terminal emulators, or you can invoke **sysadm** to provide an OSF/Motif™ interface for monitors with graphics capabilities. In either interface, **sysadm** offers extensive help for menus, operations, and queries. For more information, see the **sysadm**(1M) manual page.

Managing Disk Space

A large part of administering the DG/UX system is managing your disk space. You not only need to allocate logical disks (partitions) according to the priorities surrounding your computing needs, but you also need to keep in mind the effect that disk configurations can have on performance.

While this manual cannot tell you exactly how to allocate your disk space, it can offer tips to improve the service you get from your disks. Chapter 7, Managing Disks, covers the tools you use to manage physical and logical disks, and it discusses several service-enhancing features:

- Memory file systems, where you allocate part of your computer's physical memory to serve as a file system, thus providing fast data access.
- Software-mirrored disks, a redundant disk configuration that can make your system more robust as a file server, help safeguard valuable data, and improve performance.
- Data striping, a method of dividing up logically contiguous data elements among different physical devices in order to improve performance.
- Write verification, where the system ensures that writes to disk succeed, thus helping to protect valuable data.
- File system logging, also called "fsck-logging," a feature that allows you to have fast recovery file systems. For fast recovery file systems, the system tracks disk writes and other

file system modifications to reduce time required for recovery after disk failures or system crashes.

- Cached disk support, which allows you to use a non-volatile random access memory (NVRAM) device as an intermediate, high-performance cache to speed disk access.
- Failover disk support, which allows you to connect a physical disk or disk–array to two different systems. With a single **sysadm** operation you can transfer control of the failover disk from one system to the other, mount its file systems and start the application used to access the disk.

In addition, there are various everyday disk- and file system-related tasks:

- Copying data from disk to disk.
- Backing up and restoring data.
- Tuning file systems for improved performance.
- Searching the file system for specific files.
- Monitoring and cleaning up the file system from day to day.

Chapter 8, File System Management, covers tasks relating to file systems.

Managing System Services

There are a number of system services that the DG/UX system starts when it boots. The services become active at various run levels. See Chapter 3 for a discussion of 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. Chapters 3 discusses these services and how you can add your own.

Accommodating Users

The **sysadm** User menu, documented in Chapter 13, provides operations to control user profiles, including login directories and user groups.

To set up user terminals and lines for printers (but not the printers themselves), see Chapter 9.

The LP print service, documented in Chapter 11, provides operations for setting up printers, printer classes, forms and filters management, controlling queues and print request priorities, and starting and stopping the LP scheduler.

Starting and Stopping the System

For coverage of operations tasks such as starting (booting) and stopping (shutting down) the system, see Chapter 3.

Backing Up and Restoring Files

For information on performing archives, see Chapter 8.

Ensuring System Security

For a basic discussion of file permissions, see the manual page for **chmod**(1), which is the command used to set file and directory permissions. Chapter 8 discusses some operations the system manager can perform to check for certain kinds of security hazards. To assign user passwords and create user groups, see Chapter 13.

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 more information on the Trusted DG/UX systems, contact your Data General representative.

End of Chapter

Chapter 2
Overview of Management Tasks

This chapter briefly describes general concerns as well as the day-to-day tasks of system management. In some cases, the discussion includes hints to make the task easier.

Operating Policy

Sometimes situations arise that require you to shut down the system with little or no notice to users. Try to provide as much advance notice as possible about events affecting the use of the system. When you must take the system out of service, be sure to tell users and OS client managers when the system will be available again. The following section, "Communicating with Users," discusses the various utilities you can use to keep your users informed and alert them to system developments.

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

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
- Equipment and system configuration changes (dates and actions)
- A record of system panics and hangs
- Maintenance records (dates and actions)
- A record of recurring problems and 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 resulting from flaws in the DG/UX system software, 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, **/usr/release/dgux_5.4.2.rn**

Maintaining a User Trouble Log

As the system administrator, you can expect users to look to you to help solve any number of problems. You may find it useful to log these problems and their solutions in a User Trouble Log not unlike the System Trouble Log you may also keep. 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.

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

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, which is 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** 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**(1M) 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 get off the system so that you may perform an administrative task. For example:

```
% wall ↵
The system is going down at 3:00 for oil change — Marv ↵
<Ctrl–D>
```

In network environments, there is also the **rwall**(1C) 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**(1), **mailx**(1), and **xdgmail**(1X). 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. In a hypothetical world, Poulet wants to send a mail message to Moe. He types:

```
$ mailx moe ↵
Subject: rubber chicken ↵
Please return my rubber chicken when the puppet show is over. ↵
<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 more information on the **.mailrc** file, see the **mailx**(1) manual page. For detailed information on using mail facilities, see *Using the DG/UX™ System* and the **mail**(1), **mailx**(1), and **xdgmail**(1X) manual pages.

Boot Time Responsibilities

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**(1M) process to start various system services not provided in the kernel itself.

Depending on how you have configured your system, you may have to invoke the **init** command yourself to start the desired services. This is true if your system only boots to run level S or run level 1. 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:

```
# init 3 ↵
```

Installing the DG/UX™ System introduced you to the **init** command and run levels, so you should already know to what run level your system boots. Chapter 3 discusses run levels and the **init** command in more detail.

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

As booting occurs, a number of messages appear on the system console. 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 proceeds to configure 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 4.

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 typical array of 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**(1M), 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**, discussed in the next section.

The system only checks file systems 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 **diskman** utility (see Chapter 7), or **sysadm**'s File System menu (see Chapter 8), to start **fsck**. Also see the **fsck**(1M) manual page.

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, 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**(4) 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**(1) 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 logical disk containing the file system.

When file systems are corrupt, the **fsck** log contains messages for each error found in the file system. See Chapter 8 for a discussion of **fsck** and its output.

nfsfs.log This file contains the output from the ONC/NFS network software command that makes remote file systems available on your system. The entries in this file 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**(1) 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.

Day-to-Day Responsibilities

In addition to checking on the state of your system every time you boot, there are tasks you should perform more frequently.

Monitoring System Health: /var/adm/messages

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. To change how the system handles these messages, see "Logging System Errors and Messages" in Chapter 3 and the manual page for **syslog.conf(5).**

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 "Automating Job Execution."

Monitoring Disk Space

It is important to keep track of the size of your 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 (such as creating new files or directories or increasing the size of a file) will fail. When a file system is 90% full, only the superuser can perform an operation that involves 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`.

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.

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.

You can also use the **df**(1M) 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 logical 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 change the size of a file system, see the File System Management menu of **diskman**, described in Chapter 7.

To display the size of any directory (including any subdirectories), use the **du**(1) command. The **du** command returns the size in 512-byte blocks.

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.

Cleaning out Temporary File Directories

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 best 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 "Automating Job Execution."

Truncating /etc/wtmp

You should also reduce **/etc/wtmp** occasionally. The system logs accounting and user login information to this file on a continual basis, causing the file to grow until you reduce it. When the file becomes too large, you may choose either to remove it, replacing it with an empty **wtmp** file, or you may choose 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 **wtmp** removes information required by the accounting system to charge for connect time. See Chapter 14 for more information on the accounting system.

If you remove the oversized **wtmp** file, remember to replace it with an empty **wtmp** file. If you choose to reduce the size of the file, you should note that the **wtmp** 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**(1) 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 **wtmp** entry. If any entries in the new **wtmp** file are 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, see the **wtmp**(4) manual page.

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 more information on **cron**, see "Automating Job Execution." For more information on the LP print service logs and the **lp.proto cron** jobs, see Chapter 11.

Cleaning Up the cron Log

The **cron** utility logs a history of its activity to the **/var/cron/log** file. You should periodically truncate this file to prevent it from growing out of bounds. To truncate this file, see "Automating Job Execution."

Cleaning Up the /var/adm Directory

The **/var/adm** directory in general contains logs of various kinds that you should check occasionally. For example, if you use the system activity monitor, **sar**(1), 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 more information on these files and the accounting system, see Chapter 14.

If users on your system use the **spell**(1) 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**(1) 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 ↵
```

Cleaning Up the /var/spool Directory

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, described in Chapter 11. 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.

Cleaning Up the /var/saf Directory

On systems with a large number of users, you should occasionally check the sizes of your SAF port monitors. The logs are:

/var/saf/_log
This file contains output from the **saf** process.

/var/saf/ttymon*n*/log (where *n* is a digit between 1 and 9, inclusive)
This file contains output from a **ttymon** port monitor.

/var/saf/tcp/log
This file contains output from the **tcp** port monitor.

For more information on the SAF utility, see Chapter 9.

Making Backups of File Systems

Another important responsibility of the system manager is backing up and restoring files. If you have an operations staff at your site, you may not be the person performing the backups, but you may still want to acquaint yourself with the operations involved.

Typically, you start by backing up your entire system at the beginning of the month. Thereafter, you perform backups at the end of every workday except the last day of the week, backing up only those files that changed during that day. At the end of the last day of the week, you back up files that changed during the entire previous week. When the next month starts, you repeat the cycle.

A complete discussion of backing up and restoring file systems is beyond the scope of this chapter. See Chapter 8 for more information.

Maintaining and Verifying System Security

The DG/UX system provides 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 more information on the Trusted DG/UX systems, contact your Data General representative.

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 13, 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.

  See Chapter 8 for more information on the Check operation.
- 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 insecure script or program that has the same name as a common command.

Monitoring the Mail System

By default, the DG/UX system uses **/var/mail** as the mail directory for use by the **mailx**(1) 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.

Misaddressed mail

Typographical errors when 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. Nevertheless, 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.

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.

Superuser or postmaster mail

Some system services alert the administrator to urgent or abnormal events by sending mail to the **root** profile. Your **root** mail file may also receive messages addressed to **postmaster**, **sysadm**, or another administrative title at your site.

If you do as many administrators do and log in as a normal user, using **su** to become the superuser as needed, you may not see the **root** mail notification message that would appear at login. Therefore, it is good practice to remember to check for **root** mail occasionally, or even to 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

As system administrator, you will 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 further 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 Periodic Jobs with cron(1M)

As system administrator, 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.

The following steps detail the procedure for scheduling 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.

3. Save the file and exit from the editor. The new **cron** jobs will run at the appointed times.

   NOTE: 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

1. 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
```

2. 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.

3. 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
```

4. 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
```

Prototype Jobs for System Administration

Although the DG/UX system does not by default have any scheduled **cron** jobs, it does provide some prototype jobs that you may adapt or use as shipped. These jobs are in the following files in **/admin/crontabs** :

root.proto This 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

14), 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.

lp.proto This 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** command to become **lp** before executing **crontab –e** to schedule the desired jobs. See Chapter 11 for more information on the LP print service.

uucp.proto This 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** command to become **nuucp** before executing **crontab –e** to schedule the desired jobs. See Chapter 12 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.

Before truncating the **cron** log, stop **cron** by executing the following command line:

```
# /usr/sbin/init.d/rc.cron stop ↵
```

Then 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 ↵
```

Then restart **cron** with the following command line:

```
# /usr/sbin/init.d/rc.cron  start ↵
```

Submitting Jobs for Delayed Execution with at(1)

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 ⤶
```

NOTE: The method of rebooting shown above is not recommended for active systems. See “Shutting Down the System” in Chapter 3 for more information.

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 with batch(1)

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.

End of Chapter

Chapter 3
Operating the DG/UX System

This chapter shows you how to perform operations such as starting the system, shutting it down, recovering from trouble, collecting error messages, and dumping the system memory image and kernel file for analysis by Data General. The chapter also explains how the **init**(1M) command uses **rc** scripts to set run levels and thus provide system services.

Operational Terms

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

init	The **init**(1M) program creates all 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.
run level	Run levels, also known as run states or run modes, provide varying degrees of service on the system. These services include network-related capabilities, accounting, **cron** batch job scheduling, line printer services, and so on. Typically, run level S (for single-user mode) provides no services, and run levels 0 through 3 provide increasing levels of system functionality. By default, DG/UX provides full multiuser and network capabilities at run level 3.
rc scripts	The run command scripts are executed at every boot and every time you change run levels. At boot time or whenever run levels change, the **init** program executes scripts in a directory set up specifically for the intended run level. These scripts are the "run command," or **rc** scripts. They kill or start system services as directed by prefixes that consist of either a **K** (kill) or an **S** (start) followed by an ID number. Upon entering a run level (via the **init** command), all **rc** scripts designated in that run level are executed. Execution means that services are killed in order from highest ID number to lowest ID number. Next, processes are started in order from lowest ID number to highest ID number. The result is that only certain **rc** scripts, those that are started with the **S** switch, are active in any run level.
SAF	The SAF (Service Access Facility) manages ports, setting terminal type, mode, speed, and line discipline characteristics for the port. SAF can also start service programs for ports. An important function of SAF is to control user terminal lines, starting the **login** service for users who need to log in. For user terminal lines, SAF performs the service that the **getty** program used to provide. SAF also replaces the **uugetty** program associated with the UUCP facility.
SCM>	The System Control Monitor prompt, displayed when no operating system is running on your computer. This prompt comes from your computer hardware.

From the SCM, you use the **b** command to boot (begin execution) of your DG/UX kernel program, thereby starting the operating system. At the SCM prompt, type **h** for help, or see *Using the AViiON™ System Control Monitor (SCM)* for complete information.

DG/UX System Run Levels

You can define the run levels to provide whatever services you choose. The services themselves are controlled by the run command scripts (or **rc** scripts) 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**).

Table 3–1 shows the default run levels for the DG/UX system.

Table 3–1 DG/UX Run Levels

Run Level	Description
0	Halt the system.
i	Installation mode starts the **installman**(1M) command, which leads you through installation of the DG/UX system. At this run level, local file systems and disk services are available.
S	Single-user is a low-level run mode that is the default level the system enters upon booting. The only process running is **init**. Only the **/** (root) and **/usr** file systems are available.
1	Administrative mode is used to install and remove software utilities, run file system backups and restores, and check file system integrity. All local file systems are mounted. Only processes associated with the system console may run.
2	All local file systems are mounted. Local users can log in at terminals and use local facilities, and some out-bound network services are available. Outside systems cannot contact this system over the network. ONC/NFS services are not available.
3	The normal running mode of the DG/UX system. Complete multiuser and network services are available. ONC/NFS services are available. On workstations, the X Display Manager (**xdm**(1X)) is running.
4	User-defined run level. By default, this run level is the same as run level 3.
5	Halts the system.
6	Halts and reboots the system.
a, b, c	Pseudo run levels. These can be specified without changing a run level. Typing **init a**, for instance, invokes those entries in **inittab** that have an **a** in the **level** field. See **init**(1M).

Default Multiuser Conditions

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

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

- Local file systems are mounted in run level 2 (as they are in run level 1).
- Remote file systems are mounted in run level 3.
- The error daemon, 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 SAF (Service Access Facility) 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.

How Run Levels Are Set

This section describes how the **init**(1M) command, the **inittab**(4) file, and the **rc** (run command) scripts define a run level and determine what processes and services are available on your system.

Consider the case where you enter **init 2** to take your system to run level 2. Here is what happens:

1. You type the **init 2** command to change run levels upward from single-user mode S. You do this so you can make more services available.
2. The **init 2** command causes the **init** program to read the **inittab** file looking for all entries containing the number 2 in the *level* field. **Init** executes all lines that have 2 in the *level* field.
3. The **init** program executes all **rc** scripts associated with run level 2. These scripts perform tasks such as such as turning on accounting, starting the LP scheduler, and starting various daemons. Run level 2 is therefore defined as all those script-started processes running as a result of the **init 2** command. Output from the **rc** scripts appears in **/etc/log/init.log**.

Run Command Scripts Per Run Level

You can read the **rc** scripts to see exactly what they do. We recommend that you do not modify these scripts. You can add your own if needed. See *Porting and Developing Applications on the DG/UX™ System* for directions.

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 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 Per 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 Per 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		

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

RC Scripts in /usr/sbin/init.d

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

The check scripts are:

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

chk.fsck Runs **fsck** on all file systems listed in **/etc/fstab**. The **fsck** program is called with the **–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.

Operational Procedures

Operational procedures are necessary to keep the system running on a day–to–day basis. They include tasks such as starting and stopping the system and recovering from failure.

Starting and Restarting the System

To start the DG/UX system at the SCM prompt, use the **b** (for **boot**) command. If you set the default boot path with the SCM's **f** (for **format**) command, you can boot with this command:

```
SCM> b ⤶
```

If the default boot path is not set, you need to specify a boot path. To boot from the first SCSI disk on an AV5220 computer system, for example, 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. See the discussion of failover disks in Chapter 7 for more information.*

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**, above.

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

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

In the previous example boot command line, we loaded our standard operating system. If, for instance, you wanted to boot stand-alone **diskman**, **/usr/stand/diskman**, located on the first SCSI disk, you would execute it with a command line such as this:

```
SCM> boot sd(cisc(),0)usr:/stand/diskman ⤶
```

You can boot only those files that reside on file systems built on single-piece logical disks. For example, if your **/usr** file system were built on a logical disk made up of more than one piece, you would not be able to boot **/usr/stand/diskman**.

To reboot using different logical disks for **root** and **swap**, include the **–q** option on the **boot** command line. The command prompts you for the names of the logical disks to be used in place of **root** and **swap**. For example:

```
SCM> b sd(cisc(),1)root:/dgux –q ↵

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

If you boot a DG/UX kernel, the boot sequence begins by displaying a message that includes the boot path:

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

INIT: SINGLE USER MODE
```

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

```
SCM> b sd(cisc(),1)root:/dgux –2 ↵
```

You can use the **reboot**(1M) 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**(1M) manual page for more information.

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:
```

Changing Run Levels

You must be superuser to change run levels. You change run levels with the **init**(1M) command. Use this command line to go to administrative mode:

```
# init 1 ⮐
```

Take the system to multiuser mode:

```
# init 2 ⮐
```

Take the system to multiuser mode with network services:

```
# init 3 ⮐
```

Changing run levels causes the **init** command to run various scripts. Many of these scripts write output to **/etc/log/init.log**.

You can also use the **shutdown**(1M) command to take the system down to single-user mode.

Shutting Down the System

As superuser, you can shut down the system from the shell by changing to the root directory and using the **shutdown**(1M) command. The **sysadm** utility does not offer an operation for shutting down the system.

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**(1M) 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. See "How Run Levels Are Set" later in this chapter for details of what happens as the system comes down through various run levels.

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

Shut Down to Administrative Mode: Run Level 1

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

Options you can use are:

–y	Answers the confirmation query so 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 shut down the system:

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

Shutdown started.

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

INIT: New Run Level: 1
#
```

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

Shutting Down to Single-User Mode and Power Off

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

```
# cd  / ↵
# shutdown  -g0  ↵
```

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

```
# halt  -q  ↵

CPU HALTED


SCM>
```

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.

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 one of these failures.

Power Failure
Following a power failure, as soon as power returns to the system, it will reboot without operator intervention. See "Restoring File Systems after a Failure." If your system has the uninterruptible power supply (UPS) subsystem, see "Managing the Uninterruptible Power Supply 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" 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 "Making a Tape for Diagnosis."

If the reset switch fails to restore control to the SCM, turn off power to the computer and power the system back up. See "Restoring File Systems after a 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.

Skipping the System Dump Procedure

If you do not wish to take a system dump for submission to the Data General Customer Support Center for diagnosis, you enter **no** at the system dump prompt that appears after a panic.

NOTE: 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 set up your system to skip the system dump without operator intervention after a panic, issue the **dg_sysctl** command with the **–d** option, as follows:

```
# dg_sysctl –d skip ⤶
```

Thus configured, your system will respond to a panic by either halting or rebooting, according to how you set the automatic boot feature discussed later. If you configure your system to skip the system dump, you may skip the following section.

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. A later 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 may initiate an interactive dump by entering the following command at the SCM prompt:

```
SCM> START  1000 ⤶
```

The system then begins prompting for the required information.

NOTE: 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."

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, logical 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. To change the value of this parameter, see Chapter 4. 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)"  ↵
```

OS Client Dump Destination Device

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 Client –> OS –> Add operation. 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. The system administrator of the OS server should verify that the file system containing the dump file has sufficient free space. Use the **df**(1M) command to display the amount of free space in a file system. For more information on OS client setup, see Chapter 6.

Dumping to a Logical Disk

A system with a local disk can dump to a local logical 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 logical 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).

NOTE: You can dump to a logical disk only if it is a one-piece logical disk residing on a local SCSI disk. You cannot dump to a multi–piece logical disk or to any logical disk residing on an SMD or ESDI physical disk.

Create the logical disk with the **sysadm** operation Device –> Disk –> Logical Disk –> Create. It is a good idea to give the logical 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 logical 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 logical disk, allowing you to specify a different logical disk or a tape drive if preferred. When the system panic procedure prompts you for the dump destination device, specify the physical and logical disks using the following syntax:

ldm_dump(*physical_disk_name, logical_disk_name*)

For example, use the **dg_sysctl** command and the **–f** option to set the dump destination device to logical disk **sys_dump** on local physical disk **sd(cisc(),1)**:

```
# dg_sysctl -f "ldm_dump(sd(cisc(),1), sys_dump)" ↲
```

Rebooting after a System Failure

By default, the system halts the processor after taking a system dump, restoring control to the SCM. At this time you may reboot the system with the SCM's **BOOT** command. To minimize your system's downtime, you can set up your system to reboot without operator intervention. Set the automatic boot feature with the **dg_sysctl** command's **–r** option, as follows:

```
# dg_sysctl -r auto ↲
```

The command line above configures your system to reboot whether it performed a dump after the panic or not. 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 boot the same way now. To override this default behavior, use the **–b** option of **dg_sysctl** to specify a different boot path. For example, the following command line sets up the system to take a system dump and then boot **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 tells you of 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. Every time the system boots, it sends a notification message to any user named in **reboot_notify_ARG**.

Completing the Diagnostic Tape

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. This section tells how to finish preparing the tape. The release notice, on-line in **/usr/release**, also tells how to prepare the tape.

The 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.

NOTE: 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:

Tape If you wrote the system dump to tape after the failure, it is already on the tape as file 0.

A logical disk

If you wrote the system dump to a logical disk, transfer the system dump to a tape using the **lsd**(1M) command. For example, if your logical 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.

A file on the OS server (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 as **/srv/release/***release_name***/root/***client_name***/dgux** on the OS server. 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, 1992
File 0:  system memory dump
File 1:  kernel executable
File 2:  miscellaneous files
Density:  QIC-525 tape at high density
cpio format:  cpio –oBcv
```

Halting the System

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 ↵
```

Shutting off Power to the System

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 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**.

Restoring File Systems after a 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. See "Setup and Initialization Configuration Variables" in Chapter 4 for more information.

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 more information on fast recovery file systems and **fsck**, see the **fsck** section in Chapter 8.

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.

If a system or disk failure damages 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 the release media.

Boot stand-alone **diskman**. Use a command line such as the following one, where you specify the physical disk where the **/usr** file system resides:

```
SCM> b sd(cisc(),0)usr:/stand/diskman ⤶
```

An attempt to boot stand-alone **diskman** will fail if the **/usr** file system is corrupt or if the **/usr** file system is built on a logical disk consisting of multiple pieces.

After stand-alone **diskman** 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, you must mount the release tape and boot from it. Use a command line like this:

```
SCM> b st(cisc(),4) ⤶
```

At this point, you have two options: repair the existing (damaged) file systems, or reload the DG/UX system from the release tape.

To attempt to repair the damaged file systems, use the Check a File System operation again. Instead of specifying the **–xlp** options for **fsck** as before, specify **–y**. 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, you must re-install the DG/UX system completely. Do this by going to the Logical Disk Management Menu and using the Delete a Logical Disk operation to delete the logical disks containing the **/** and **/usr** file systems (**root** and **usr** logical disks). Then go to the System Installation Menu and select option 5, All Installation Steps. You will also need to re-install any packages that had been installed before the failure.

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 System Installation Menu and performing the Load System Software operation. The Load System Software operation will load system files as necessary. When it has finished, you may replace any files you wish by restoring them from backup.

Logging System Errors and Messages

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

```
selector    action
```

The `selector` field contains a list of priority specifications separated by semicolons:

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

where `facility` is the origin of logged messages, such as a user or **mail**. Possible values for `facility` are:

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

mail	The mail system.
daemon	System daemons such as **routed** and **ftpd**.
auth	The authorization system: **login**, **su**, and **ttymon** (which replaces **getty** and **uugetty**, among other things).
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.
ups	The uninterruptible power supply subsystem (with supporting hardware only).
uucp	Reserved for the UUCP system.
*	An asterisk indicates all facilities except the mark facility.

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

emerg	For panic conditions that would normally be broadcast to all users.
alert	For conditions that should be corrected immediately, such as a corrupted system database.
crit	For messages about critical conditions, such as hard device errors.
err	For other errors.
warning	For warning messages.
notice	For conditions that are not error conditions, but may require attention.
info	Informational messages.
debug	For messages that are normally used when debugging a program.
none	Means do not send messages from the indicated facility to the selected file. For example, a selector of `*.debug;mail.none` will forward all messages except mail messages.

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

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

Licensed Material – Property of copyright holder(s)

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

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

*.emerg                                               *
```

The **rc.syslogd** script starts **syslogd**.

Managing the Uninterruptible Power Supply Subsystem

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.

Specifically, the UPS subsystem functions as follows. 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**. See **syslog.conf**(5) for more information.
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.

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. 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 Device -> Port -> Port Service -> Delete. See Chapter 9 for more information on port services.

NOTE: 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.

NOTE: If you start the UPS daemon before you have installed the UPS backup battery unit, unpredictable results could occur.

Execute the **sysadm** operation Device -> 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 Chapter 9 for more 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 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 Device –> Port –> Terminal –> Add to start the **login** service for the serial port.

Setting and Displaying UPS Parameters

The menu Device –> UPS –> Params 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 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.

Expert Run Level Information

You do not have to read this expert section to operate the DG/UX system. Information here is optional and is provided to enhance your understanding of how run levels work. For basic information about run levels, see "DG/UX System Run Levels" earlier in the chapter.

The Fundamentals

The **inittab** file contains entries specifying which processes will be invoked at which run level.

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

The **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 **rc** scripts. Commonly called "run command" scripts, they 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 run command script in **init.d**.

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

The Sequence

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

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

   ```
   # init 3 ↲
   ```

2. The **init** program scans the **inittab** file for all entries containing the run level number 3 in the *run level* field. Then, **init** invokes the `rc.init 3` instruction which is in the *process* field.
3. The **/sbin/rc.init** program uses the run level number 3 as a pointer to directory **/etc/rc3.d**, which contains links to the scripts in **/usr/sbin/init.d**. A script called **rc.syslogd** starts the **syslogd** program.
4. The **rc.init** program then executes all scripts for run level 3; among these is **syslogd.**

The /etc/inittab File

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

`id:level:action:process`

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 only 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 pound sign (#). The **init** program ignores everything appearing after a pound sign on a line.

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

```
#
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 3–1 The Prototype /etc/inittab File

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**(4).

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, described earlier in the chapter.

RC Scripts and Check Scripts

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

chk.*	These scripts are usually run once, at boot time. An example is **chk.fsck** which runs the **fsck** program on file systems.
rc.*	These scripts are invoked with either a start or stop argument. An example is **rc.tcload**, which starts or stops the asynchronous terminal I/O controllers.

Init.d Links

Typically, the above scripts exist in **/usr/sbin/init.d**. The **rc** scripts are invoked via links in an **/etc/rc*N*.d** directory. Remember, 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:

`S`*`nnn.name`*

or

`K`*`nnn.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 3–2 is a portion of 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.update          113  K   K   S   S   S   S   K   K   S
    rc.localfs         114  K   K   S   S   S   S   K   K   S
    rc.sync            116  K   K   K   S   S   S   K   K   -
    rc.lan             117  K   K   K   S   S   S   K   K   -
    rc.setup           119  K   K   S   S   S   S   K   K   -
    rc.daemon          130  K   K   S   S   S   S   K   K   -
    rc.install         131  -   -   -   -   -   -   -   -   S
##  special systems    150

    rc.usrproc         210  K   K   K   S   S   S   K   K   -
##  rc.mcli            211  K   K   K   S   S   S   K   K   -
##  rc.eventd          2111 K   K   K   S   S   S   K   K   -
    rc.llc             212  K   K   K   S   S   S   K   K   -
##  rc.omtran          214  K   K   K   S   S   S   K   K   -
##  rc.netbeui         216  K   K   K   S   S   S   K   K   -
##  rc.x25port         217  K   K   K   S   S   S   K   K   -
##  rc.tsp             230  K   K   K   S   S   S   K   K   -
##  rc.tcpipport       232  K   K   K   S   S   S   K   K   -
    rc.syslogd         235  K   K   K   S   S   S   K   K   -
    rc.dgserv          236  K   K   K   S   S   S   K   K   -
```

Licensed Material – Property of copyright holder(s)

```
##  rc.ypserv          237  -   -   -   S   S   S   -   -   -
##  rc.nfslockd        239  K   K   K   S   S   S   K   K   -
##  special systems    250
    rc.account         251  K   K   K   S   S   S   K   K   -
    rc.cron            252  K   K   K   S   S   S   K   K   -
    rc.lpsched         253  K   K   K   S   S   S   K   K   -
    rc.preserve        254  K   K   -   S   S   S   K   K   -
##  rc.nbvt            255  K   K   K   S   S   S   K   K   -
##  rc.nw_tran         270  K   K   K   S   S   S   K   K   -
##  rc.snalan          271  K   K   K   S   S   S   K   K   -
##  rc.sdlc            272  K   K   K   S   S   S   K   K   -
##  rc.icobol          293  K   K   K   S   S   S   K   K   -

##  rc.omserv          314  K   K   K   K   S   S   -   -   -
##  rc.nfsserv         315  -   -   -   K   S   S   -   -   -
##  rc.x25serv         317  -   -   -   K   S   S   -   -   -
##  rc.osit            318  K   K   K   K   S   S   K   K   -
##  rc.alp             319  K   K   K   K   S   S   K   K   -
##  rc.icl             330  K   K   K   K   S   S   K   K   -
##  rc.x400            331  K   K   K   K   S   S   K   K   -
##  rc.lanman          332  K   K   K   K   S   S   K   K   -
##  rc.tcpipserv       334  -   -   -   K   S   S   -   -   -
##  rc.smtpgw          336  K   K   K   K   S   S   K   K   -
##  special systems    350
##  rc.nfsfs           353  K   K   K   K   S   S   K   K   -
##  rc.upsd            355  K   K   K   K   S   S   K   K   -
##  rc.upsd.client     356  K   K   K   K   S   S   K   K   -
    rc.failover        358  K   K   K   K   S   S   K   K   -
##  rc.nw_serv         370  K   K   K   K   S   S   K   K   -
##  rc.sna             371
##  rc.sna             372
##  rc.sna             373
##  rc.x500            375
##  rc.cmips_agent     378
##  rc.oseyenode       379  K   K   K   K   S   S   K   K   -
##  rc.x11             391
##  rc.dni             395  K   K   K   K   S   S   K   K   -
##  rc.dims            396  K   K   K   K   S   S   K   K   -

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

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

Figure 3–2 RC Scripts: the Kill and Start Mechanism

You can think of all **rc** scripts as being either on (S) or off (K). Since TCP/IP and ONC/NFS are optional products, we show their links commented out above. The **/etc/dgux.rclinktab.proto** file contains comments explaining this table.

Changing the Behavior of RC Scripts

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

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

To add your own **rc** scripts, see *Porting and Developing Applications on the DG/UX™ System.*

End of Chapter

Chapter 4
System Configuration Management

The first part of this chapter lists the administrative logins for the DG/UX system and covers the following procedures:

- Recovering from a forgotten **root** or **sysadm** password
- Monitoring system activity
- Managing the accounting system
- Monitoring system process activity
- Managing security databases
- Building and executing a kernel
- Setting system parameters
- Setting the language environment
- Setting system date and time

The second part of this chapter offers suggestions for performance management and concludes with listings and definitions of the tunable parameters for the DG/UX system.

The DG/UX Administrative Logins

On the DG/UX system, a file's owner controls access to the file. The superuser logins, **root** and **sysadm**, which exist on all systems, can override any permission settings and can execute, open, read, delete, or change any file in the system. If you know the **root** or **sysadm** password, you can become the superuser by executing the **su**(1) command with the – (hyphen) option followed by the desired login name. We recommend that you use **sysadm** rather than **root**, like this:

```
% su - sysadm ⮐
```

When you log in as **sysadm**, your home directory is the **/admin** directory instead of **/**. Using this login keeps your personal administrative files out of **/**. As an alternative to using **su** to become **sysadm**, you can simply log in as **sysadm**.

The DG/UX system disperses system file ownership over 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 **/etc/passwd**, you'll see that the system logins have an asterisk (*) in the password field, meaning that no one can log in with these.

Generally, you should perform administrative tasks as **sysadm**. In some cases, however, you may need to use **su** to change to another system login name for some operations. For example, you should become **nuucp** before changing the **cron** jobs for UUCP. You do not want to become **uucp**, however, because that login is reserved for use by the UUCP facility's **uucico** daemon. For more information on UUCP, including a discussion of the difference between the **uucp** and **nuucp** logins, see Chapter 12. Table 4–1 shows the administrative and system logins.

Table 4–1 Default DG/UX 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**(1X), the **xdm** control utility. See the **telxdm**(1X) 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	The system administrator 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 4–1, only those entries without asterisks may be used as actual logins; the others are for system use only. The network packages may add other logins to the **/etc/passwd** file.

Recovering Forgotten Superuser Password

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**(1) 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 **wall**(1M) 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 diskman, either from disk or from the release tape, and check the root file system. If you you need to repair damaged DG/UX system files, see Chapter 3.
5. Once you are in single–user mode, assign new **root** and **sysadm** passwords with the **passwd**(1) command.

Monitoring System Activity

The DG/UX system has a system activity reporting mechanism, **sar**(1), that you can use to review statistics on CPU performance, disk and terminal I/O, memory usage, process communication and execution, 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.

The System Activity menu provides operations for starting and stopping **sar**, deleting old data collections, and reviewing reports.

The **sar** utility accumulates the data and produces reports. It does not run all the time, so you have to start it and let it run for a while in order to get data to review. When **sar** runs, it accumulates data on all system activities, sampling system data at regular intervals until it has produced a given number of samples. When you start data collection, you may specify the interval between samples and the number of samples. When you display the data, you may specify which types of data to review, or you may review all of it. For complete information on the system activity monitor, see **sar**(1) and **sar**(1M).

Starting System Activity Monitoring

Select the Start operation 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 before stopping. Keep in mind that each sample is over 1,000 bytes in size.

The system monitor stores the data in the file **/var/adm/sa/spd.***name*. The file remains until you delete it explicitly with the shell's **rm**(1) command or with the Delete operation.

You may run multiple system monitoring sessions at the same time.

Stopping System Activity Monitoring

To halt a data collection session before the monitor has collected the specified number of samples, select the Stop operation. 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 Delete operation 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

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, which is 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).

Monitoring Process Activity

Select the 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 the Delete operation 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

To change the priority of a running process, select the Modify operation. 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 List operation 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.

Signaling Processes

Use the Signal operation to send a signal to one or more processes. The effect of the signal depends on the receiving process. To kill a process, use the Delete operation, 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.

Building and Booting a Kernel

The kernel is the 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**.

Although the DG/UX system ships with a starter kernel, you 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.

To help you manage your system's kernel, the System menu includes the Kernel menu, which provides these operations:

`Auto Configure`
: Build a kernel that includes default parameter assignments and support for all hardware devices installed at standard locations.

`Build`
: Build a kernel that is the same as the one built with Auto Configure except that you can edit the system file and customize the tunable parameter assignments and add entries for nonstandard devices.

`Reboot`
: Shut down the system completely (except for the hardware itself) and restart the operating system.

The following sections elaborate on these operations.

Building a Kernel

You build a kernel with the Auto Configure or Build operation. 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 "Tuning System Parameters" at the end of this chapter.

probedev 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, called **probedev**(1M), which 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 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**.

If you do not wish to tune your kernel at this time, 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.

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. If your system is one of these hybrid configurations, see *Customizing the DG/UX™ System* for more information.

Whether you build a kernel now with Auto Configure or with Build, you may build a new kernel with either operation at any time.

Building a Kernel with Auto Configure

The Auto Configure operation starts by presenting you with these queries:

If your system has exclusive access to a H.A.D.A. II subsystem (that is, your host is the only one using disks in the H.A.D.A. II subsystem), you may find the Auto Configure operation helpful for building your kernel. If two hosts use disks in the same H.A.D.A. II subsystem, however, you should use Build to build your kernel.

`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 following 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 Kernel with Build

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.

The Build operation presents the following queries:

`System configuration file name`

This name distinguishes the system file and the kernel from existing system files and kernels. The system file is **system.***name*, located 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 lets you edit the file before performing the build. The operation does not generate a new list of installed devices for the system file; therefore, if you have installed new devices, you have to add the entries to the file yourself.

If you enter a new name for a system file, the operation creates it by:

1. Invoking **probedev** to generate a list of installed standard devices. The list goes at the beginning of the file.
2. Concatenating the existing prototype system files, named **system.*package*.proto** in **/usr/src/uts/aviion/cf**, to form the body of the new system file.

`Build for this host or for OS client(s) of this host`

Select `this host` for the normal case, where you are building a kernel for this system.

Select `OS client(s)` if you are building a kernel for use by one or more OS clients. The remainder of this section describes the implications of these options in more detail.

If your system is a hybrid system, one that has a local disk but gets all or part of its operating system services from an OS server host, see *Customizing the DG/UX™ System* for more information.

`Editor`

Enter the full pathname of the editor that you wish to use. The default is **vi**(1).

Next, the operation starts the selected editor and lets you edit the system file. The system file consists of prototype files from the various software packages installed on your system, concatenated to form one large file.

The system file 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 entries of particular interest to you are:

Hardware devices

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(),*)
```

CAUTION: *If building a kernel for a system in a dual-initiated configuration (one where two systems share a SCSI bus), do not use the asterisk notation described above. If two systems sharing a SCSI bus configure the same device at boot, a SCSI bus race condition will occur, and neither system will be able to access any devices on the bus.*

If you intend to use MS-DOS® file systems, you need to add the MS-DOS file system manager driver, **dfm()**. See **dfm**(4) for more information.

If you intend to use High Sierra or ISO 9660 CD-ROM file systems, you need to add the High Sierra file system manager driver, **hfm()**. See **hfm**(4) for more information.

If you are building the kernel for an OS client, the system file contains three tunable parameters at the beginning. These parameters are necessary for the system to function as an OS client. If you are building the kernel for this host, the system file does not contain these tunable parameters. If you want this host to function as an OS client, you must add these parameter declarations:

```
NETBOOTDEV          "inen()"
ROOTFSTYPE          NETWORK_ROOT
SWAPDEVTYPE         NETWORK_SWAP
```

Other than these three parameters, the tunable parameters are the same whether building for an OS client or for this host.

To review the entire set of available parameters, see the text files in **/usr/etc/master.d**. This information determines what devices the DG/UX system can recognize and what values a number of system parameters will have. 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.

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

For more information on tuning parameters, including a discussion of the parameters that you are most likely to want to change, see "Tuning System Parameters," later in the chapter.

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`

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.

If you choose not to create the link, the existing kernel remains linked to **/dgux**.

To build the kernel, the operation first runs **config**(1M) on the system file and produces program code in a file named **conf.c**. If **config** fails, see "Configuration Error Messages." Correct the problem and invoke Build again. After **config** succeeds, 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 ↵
```

A new kernel takes effect the next time the system boots.

For more information on installing and setting up OS clients, see *Customizing the DG/UX™ 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.

Configuration Error Messages

The following error messages are generated by the **config**(1M) program. Some errors originate in the master file, others in the system file. Errors in the system file are more common since you change it as a result of updating your configuration. Errors in the master file are less common; normally you 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.

Setting and Displaying DG/UX Parameters

The 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. The parameters also control a number of other things.

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**(1) command to set the date and time while the system is running.

NOTE: 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**(1M), 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 more information on the system error logger, see Chapter 3.

`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. See Chapter 14 for more information on the accounting system. See "Automating Job Execution" in Chapter 2 for more information on **cron**.

`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. See "Automating Job Execution" in Chapter 2 and the **cron**(1M) manual page for more information on **cron**.

`Start lpsched Printer Scheduler at Boot`

Select this feature to start the **lpsched**(1M) printer scheduler daemon each time the system is booted. This daemon must be running in order for users to print jobs with the **lp**(1) command. This daemon also supports print requests submitted with the **lpr**(1) command. If not started at boot time, you can start this daemon later with **sysadm**(1M).

`Start lpd Printer Scheduler at Boot`

Select this feature to start the **lpd**(1M) printer scheduler daemon each time the system is booted. This daemon supports print requests submitted with the **lpr**(1) 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**(1) or **vi**(1) 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 readahead, most file buffer writebacks, 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**(1M) command. The **swapon** command runs each time you boot the system, checking the **/etc/fstab** file for entries of type **swap** and making those logical 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 logical 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 logical-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**(1M) for more information. Refer to the file **/etc/devlinktab** to see how logical 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**(1) and **ex**(1) 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 parameters 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**(1M) command. Once set, write-verify defaults appear in the **/etc/default/dkctl** file. The default is **true**. For more information on the write-verify feature, see Chapter 7.

fsck_ARG
: This parameter provides the arguments to be supplied to the file system checker, **fsck**(1M), 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**, below. 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**.

Other parameters also appear in **/etc/dgux.params**. To change them, edit the file. The file contains comments to help you understand the parameters and the accepted values.

Setting Global Locale Variables

The Language menu provides operations for setting and listing the native language support parameters for your system.

The native language parameter, **LANG**, tells the system which locale database to use for determining functionality that varies from region to region throughout the world. For example, the locale database determines currency symbols, collating sequences for sorting operations, comma and decimal point usage and placement for numerals, and so on. Certain DG/UX commands, including **cp**(1), **find**(1), **ln**(1), **mv**(1), **rm**(1), and **tar**(1), accept native language responses to yes/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 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. 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.

This release of the DG/UX system includes locale databases for 70 different locales. A locale does not necessarily correspond to a single language, country, or geographic region. Locale names, which are the values you may assign to the **LANG** variable, have the form

```
language [_territory [.codeset ]]
```

Some examples are **fr** (French), **fr_CA** (Canadian French), and **fr_CA.850** (Canadian French, using the IBM 850 code set). The code set normally available 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 codeset 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 locale definitions supplied as part of the DG/UX system appear in the following table. These are the values that you may assign to the **LANG** variable. These all appear as subdirectories of **/usr/lib/locale**. You are free to delete any that you do not need; however, you must leave the **C** locale definitions intact. The contents of the locale definitions are described in the **setlocale**(3C), **colltbl**(1M), **chrtbl**(1M), **montbl**(1M), and **mkmsgs**(1) manual pages.

Table 4-2 shows the locale names and related information for locales supported on the DG/UX system.

Table 4-2 Supported Locales

Locale Name	Language	Country	Code Set	Equivalent Name
C	English	US	ASCII	C-locale
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.646	English	United Kingdom	ISO 646	
en_GB.437	English	United Kingdom	PC 437	
en_GB.850	English	United Kingdom	PC 850	
en_AU	English	Australia	ISO 8859-1	
en_AU.646	English	Australia	ISO 646	
en_AU.437	English	Australia	PC 437	

Continued

Table 4–2 Supported Locales

Locale Name	Language	Country	Code Set	Equivalent Name
en_AU.850	English	Australia	PC 850	
en_CA	English	Canada	ISO 8859–1	
en_CA.646	English	Canada	ISO 646	
en_CA.437	English	Canada	PC 437	
en_CA.850	English	Canada	PC 850	
en_CA.863	English	Canada	PC 863	
en_US	English	United States	ISO 8859–1	
en_US.646	English	United States	ISO 646	
en_US.437	English	United States	PC 437	
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

Continued

Table 4–2 Supported Locales

Locale Name	Language	Country	Code Set	Equivalent Name
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	
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	USSR	ISO 8859–5	ru_SU
sh	SerboCroatian	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	
sk	Czechoslovakian	Czechoslovakia	ISO 8859–2	

Setting Time and Date

To set the time and date, use the System –> Date –> Set operation. When you set the time, you may specify:

- Month
- 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**(4) manual page for more information on the time zone format.

We recommend that you do 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 System –> Date –> Get operation. In the shell, you set and display the date and time with the **date**(1) command.

Improving Performance

This section contains suggestions for improving the performance of your system. You may want to reread the planning sections in *Installing the DG/UX™ System.*

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

Maximizing System Usage

To ensure maximum system performance, you should check for:

- Less important jobs interfering with more important jobs
- Unnecessary jobs
- Jobs running during peak hours that could just as well run during off–peak hours
- The efficiency of user–defined features, such as personal configuration files (**.profile** and **.login**) and the PATH environment variable

Getting Process Information

To obtain information about active processes, use the 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 "runaway" process, one 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" in Chapter 2 for more information on these commands.

Checking User Search Path Variables

Every shell process has a **path** or **PATH** environment variable that lists the directories that the system should search when looking for a command invoked by the user. Every time the user issues a command, the system scans the directories in the path to see where the command resides. If the command invokes other commands, the system has to scan the search path for them too. These searches require both processor and disk time, thus changes here can help performance.

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

Path Efficiency.
The system searches the path directories in the order listed, so your most commonly used directories should appear first in the path. Make sure that a directory is not searched more than once for a command.

Convenience and Human Factors.
Users may prefer to have the current directory listed first in the path (**.:/usr/bin**), but note that putting the current directory first can lead to breaches in security. If a program having the same name as a system command, such as **ls** or **pwd**, for example, is in the user's current directory, the user will inadvertently execute it while intending to execute the "real" system command. Depending on the nature of the "fake" system command that exists in the local directory, the results could be undesirable, or at best, unpredictable.

It is particularly critical that the superuser profiles (**root** and **sysadm**) not have the current directory first on their path. In fact, it is safer to leave the current directory out of a superuser path altogether.

Path Length.
In general, the search path should have the least number of required entries.

Large Directory Searches.
Avoid searching large directories if possible. Put any large directories at the end of the search path.

Shift 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 14) 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, noninteractive jobs (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.

Tuning System Parameters

Tunable system parameters set various table sizes and system thresholds to handle the expected load on your system. You'll find the default tunable parameter values are adequate for most configurations and applications. If your application has special performance needs, you may have to experiment with different combinations to find an optimal set. To set tunable parameters, edit the values in your system file when building a new kernel with the Build operation.

Uname Configuration Variables

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

NODE The UUCP nodename of the system (**sales**, **sys31**, and so on). The default is **no_node**.

MACH The name of the system's underlying hardware. The default is **AViiON**.

SYS The name of the operating system. The default is **dgux**.

REL The number of the system's release. The default for this release is **5.4.2**.

VER The version number of the operating system. The default is **generic**.

ARCH The instruction set architecture of the computer hardware. The default is **mc88100**.

HW_PROVIDER
Name of the company that built the computer hardware. The default is **Data General**.

Setup and Initialization Configuration Variables

The setup and initialization configuration variables are also listed in **/usr/etc/master.d/dgux**. These variables set the system initialization parameters shown in the following list.

DST The type of Daylight Savings Time being used. The different types are defined in **/usr/include/sys/time.h**. The default is **1**.

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

DUMP The name of the default system dump device in DG/UX common device specification format. This device is the default device used to do a system memory dump after a panic or a halt. The default is **st(insc(),4)**.

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

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

INIT The internal function that is executed upon system booting. The default function calls the program specified in the INITPATH parameter. Do not change this parameter. The default is **&init_run_sbin_init**.

INITPATH
The system initialization program that the kernel should execute as the very first user process on the system. The default is **/sbin/init**.

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

SHOWGOODCONFIGS
A boolean indicating whether or not the system will print a message for every successful attempt to configure a device during system initialization. Use **1** for true and **0** for false. The default is **0**.

SHOWBADCONFIGS
A boolean indicating whether or not the system will print a message for every unsuccessful attempt to configure a device during system initialization. Use **1** for true and **0** for false. The default is **1**.

RUNFSCK
A boolean indicating whether or not the kernel should run **fsck** to check the root file system before trying to mount it. Use **1** for true and **0** for false. The default is **1**.

FSCKFLAGS
The options to be used when invoking **fsck** to check the root file system before attempting to mount it. This parameter is only relevant when RUNFSCK is true. The default is **–xlq**.

CPU and Process Configuration Variables

The CPU and process configuration variables are also listed in **/usr/etc/master.d/dgux**.

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

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

MAXSLICE
Specifies the maximum time in milliseconds a user process can run before being

suspended. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE number of milliseconds. The default is **500** (1/2 second).

MAXUP
Specifies the maximum number of processes that a nonsuperuser can have in existence at one time. The default is **50**. This value should not exceed the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if ten people logged in with the same user ID, the default limit would be reached very quickly.

MAXBUFAGE
Specifies the maximum age in seconds that a modified buffer in main memory can reach before it is written to disk. The default is **60**.

MAXSYSBUFAGE
Specifies the maximum age in seconds that a modified buffer in main memory containing system data can reach before it is written to disk. If this parameter is **0**, the default, the system uses the same maximum age value for system buffers that it uses for user buffers (as set in MAXBUFAGE). If you set MAXSYSBUFAGE to a value greater than MAXBUFAGE, the system ignores it and uses MAXBUFAGE for all buffers.

NVPS Specifies the number of virtual processors available for user processes. If set to **0**, the default, the system determines the number of virtual processors based on the amount of physical memory and the value of NPROC. The system never uses more than NPROC number of virtual processors.

PERCENTSYSBUF
Specifies the percentage of system memory (after initialization) that is reserved for system buffers. System buffers hold directory, inode, file index, and bitmap data. The default, **0**, causes the system to select a reasonable value for the system.

Pseudo–Device Unit Count Variables

The pseudo–device unit count variables set the number of units a specified pseudo–device will have. (No count variables are needed for real devices; any units present are usable.)

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

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

WMTCOUNT
The number of pseudo–magnetic tape devices used for access to WORM (write–once/read–many) devices (**/dev/wmt/***). The default is **8**.

File System Configuration Variables

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

ACCTON, ACCTOFF

If the free space in the file system in which the accounting file resides becomes less than the percentage specified by ACCTOFF, no further accounting records will be written. When the free space reaches ACCTON percent, the writing of accounting records will resume. ACCTOFF should always be smaller than ACCTON. The default for ACCTON is **5**. The default for ACCTOFF is **2**.

CDLIMIT

Specifies the maximum size in bytes that a nonsuperuser file may attain. The default is the constant **INT32_MAX**, which is equal to 2,147,483,648.

FREEINODE

Specifies the maximum ratio of in–use inodes to free inodes in the system. To improve performance on systems where you open a large number of files repeatedly, set this parameter to a higher value. The default is **4**.

FREERNODE

Specifies the maximum ratio of in–use rnodes to free rnodes in the system. To improve performance on systems where you open a large number of files repeatedly, set this parameter to a higher value. The default is **4**.

ROOTLOGSIZE

Specifies the size in 512–byte blocks that the fast recovery **fsck** log for the root file system should be. If **0**, the default, the root file system is not mounted as a fast recovery file system. For more information on fast recovery **fsck**, see Chapter 8.

STREAMS Configuration Variables

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

PERCENTSTR

Specifies the percentage of system memory (after initialization) that is reserved for STREAMS buffers. The default is **20**.

STRLOFRAC

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

STRMEDFRAC

Specifies the threshold percentage of in–use STREAMS buffers beyond which medium–priority requests for STREAMS buffers will be denied. This parameter must always be in the range **STRLOFRAC <= STRMEDFRAC <= 100**. The default is **90**.

NQUEUE

Specifies the maximum number of STREAMS queues (Streams plus instances of STREAMS modules) that may exist at any one time on the system. A minimal stream contains two queue pairs: one for the Stream head and one for the driver. Each instance of a module on a Stream requires an additional queue pair. The default is **2048**.

If the system returns the error **SFM_ENOSR_QUEUE_LIMIT**, it has reached the limit on STREAMS queue pairs, and you need to set **NQUEUE** to a higher value.

NSTRPUSH

Specifies the maximum number of STREAMS modules that may be pushed on any one Stream. This is used to prevent an errant user process from consuming all the available queue pairs on a single STREAMS module. The default is **9**.

STRMSGSZ

Specifies the maximum number of bytes allowed in the data portion of a STREAMS message. A module maximum packet size of **INFPSZ** (defined in **/usr/include/sys/stream.h**) defaults the maximum packet size to this value. The default is **4096**.

If **STRMSGSZ** is set to zero, the per module/driver **maxpsz** is used. Also, if **STRMSGSZ** is set to zero and the module/driver **maxpsz** is **INFPSZ**, no limit is imposed. If **STRMSGSZ** is greater than zero, it specifies a global limit such that any per module/driver **maxpsz** that is greater than **STRMSGSZ** is ignored in favor of the global **STRMSGSZ** limit.

If **STRMSGSZ** is larger than necessary, a single **write** or **putmsg** can consume an inordinate number of data blocks.

STRMCTLSZ

Specifies the maximum number of bytes allowed in the control portion of a STREAMS message. The control part of a message created with **putmsg** is not subject to the constraints of the minimum or maximum packet size, so this value is the only way of providing a limit for the control part of a message. The default is **1024**.

Semaphore Configuration Variables

These semaphore configuration variables are also listed in **/usr/etc/master.d/dgux**. The variables are:

SEMMNI

Specifies the maximum number of unique semaphore sets that may be active at any one time on the system. The default is **1024**.

SEMMSL

Specifies the maximum number of semaphores that a semaphore set may contain. The default is **256**.

SEMOPM

Specifies the maximum number of semaphore operations that can be executed per **semop**(2) system call. The default is **10**.

SEMVMX

Specifies the maximum value a semaphore may have. The default is the maximum value for this parameter, **32767**.

SEMUME

Specifies the maximum number of undo entries per undo structure. The default is **10**.

SEMAEM

Specifies the maximum value of the adjustment for adjust–on–exit. The value is used whenever a semaphore value becomes greater than or equal to the absolute value of **semop**(2), unless the program has set its own value. The default value is the maximum value for this parameter, **16384**.

SEMAPM

The maximum number of processes that may specify adjust–on–exit at one time. The default is **16384**.

Shared Memory Configuration Variables

The tunable parameters shown below are associated with interprocess communication shared memory. These parameters are also defined in the **/usr/etc/master.d/dgux** file.

SHMMNI

Specifies the maximum number of shared memory identifiers system wide. Each entry contains 52 bytes. The default is **1024**.

SHMSEG

Specifies the number of attached shared memory segments per process. The default is **256**.

SHMMAX

Specifies the maximum shared memory segment size in bytes. The default is **4*1024*1024**.

SHMMIN

Specifies the minimum shared memory segment size. The default is **1**.

Message Configuration Variables

These parameter variables are also listed in **/usr/etc/master.d/dgux**. They set the message parameters shown in the following list.

MSGMNI

Specifies the maximum number of message queues that may exist in the system at one time. The default is **1024**.

MSGTQL

Specifies the maximum number of outstanding messages that may exist in the system at one time. The default is **1024**.

MSGMNB

Specifies the maximum number of bytes that a message queue may contain. The default is **4096**.

MSGMAX

Specifies the maximum number of bytes that a message may contain. The default **2048**.

End of Chapter

Chapter 5
Managing Software

The software packages on your system fall into the following categories:

- Operating system releases for OS clients, which are loaded into release areas.
- 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.

The **sysadm** utility provides a menu and operations for each category.

Managing Release Areas

A release is a directory environment, or *release area*, that contains software that provides operating system service for OS clients. We distinguish between the *primary* release and *secondary* releases. The primary release runs as the operating system on OS servers and stand–alone systems. Any other releases are secondary releases.

For each release installed on your system, there is a directory in the **/srv/release** directory structure. For the primary release, there is **/srv/release/PRIMARY**, and for each secondary release, **/srv/release/***release_name*. 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 Software –> Release Area menu of **sysadm** contains the operations you use to create, delete, and list release areas. The following sections elaborate on these tasks.

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 6 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, 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 logical disks, creating file systems on them, and mounting them in some appropriate place under **/srv/release**.

For example, if you were going to add a release called **dgux4.30**, you would need to make sure that **/srv/release/dgux4.30** had enough space for:

- one copy of the DG/UX 4.30 system's **/usr** file system and
- the OS clients' root file systems.

You could create and mount a file system at **/srv/release/dgux4.30/usr** and another at **/srv/release/dgux4.30/root**.

Chapter 7 explains how to create logical disks. Chapter 8 explains how to create, add, and mount a file system.

The **sysadm** utility provides an operation for creating a release area, but before you invoke it, 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. See Chapter 6 to add OS clients.

Deleting a Release Area

To delete a release area, use the 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 List operation. You can display information on all releases or on a specific release.

Package Management

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 logical 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 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 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.

NOTE: 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 MB.

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 a multi-piece logical disk; therefore, do not expand the **/** (root) or **/usr** file systems if doing so would add a piece to the underlying logical disk. If you need to add space to **/** or **/usr**, first make sure that the desired amount of free space exists on the physical disk immediately after the logical disk you wish to expand.

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 MB software package, allocate 59 MB of disk space.

See Chapter 8 for information on expanding file systems.

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.

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.

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, use the 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 desire access to the package will have to mount the new file systems onto the appropriate directory on their system.

Setting Up Software in a Release Area

To set up software that you have already loaded, use the operation Software –> Package –> Set up.

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.

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.

Listing Packages

To display information about software packages on release media or installed on your system, select the 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.

88open Package Management

In addition to installing packages that conform to the DG/UX software package guidelines, you may 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 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 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 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.

Applications Management

The Applications Management menu provides menus and operations for any additional software that you have installed on your system beyond just 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.

End of Chapter

Licensed Material – Property of copyright holder(s)

Chapter 6
Managing Clients

This chapter tells how to manage OS (operating system) and X terminal clients. An OS client is a workstation that depends on another system, called the OS server, for its operating system software. 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 OS client operations apply only if you are the administrator of an OS server system.

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 a server in somewhat the same way that an OS client does,

The **sysadm** Client menu provides menus containing operations for managing OS and X terminal clients. Of the two main sections of this chapter, the first section covers the OS Client menu, and the second section covers the X Terminal menu.

Managing OS Clients

The 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.

Adding an OS Client to a Release

You should create a kernel for an OS client before adding the OS client. Use the System –> Kernel –> Build operation, discussed in Chapter 4.

To add an OS client, select the Add operation. You may attach a client to multiple releases by performing the Add operation once for each release. Use the Set operation, discussed later, to select which release the OS client boots.

After a release has been set up on a server (see the **sysadm** Software menu), you can add an OS client to that release. Adding an OS client means creating a root directory for the client and then recording information about the client. Here are the steps for adding an OS client:

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 –> Hosts –> Add. Get the Internet address from your network administrator. Then add the client's Ethernet address to the **ethers** database with the operation Networking –> TCP/IP –> 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 as noted:

   /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.

   /srv/dump
   : This directory is for memory dump images that OS clients may produce in the event of a system panic or any time the client administrator chooses to make a memory dump. 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 the above directories are not large enough, you may need to increase the size of the file systems or create new file systems. See Chapter 7 for more information on increasing the size of a file system and on creating logical disks and file systems. Chapter 8 tells how to add file systems to the file system table and how to make file systems accessible.

4. Perform the Add operation for each client.
5. Set up software packages on the client system. See the Software menu.

NOTE: 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 and/or your Data General representative.

The Add operation requires that you supply the following information:

`Client Host Name`
: This is the name by which the OS client is known on the net. 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 MB (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, 16 MB, is about 32,000 512–byte 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 MB (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. The size of the swap file does not vary; it is always the size allocated.

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 more information on NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

Client Kernel Link

An OS client boots **/srv/release/***release_name***/root/_Kernels/dgux.diskless** 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 4.

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/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 a Client from a Release

Select the Delete operation 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 a Client's Bootstrap Link

Select the 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 Client Information

You can use this operation in two ways. You can display general information about all clients by specifying **all** (the default), or you can display detailed information about a single client.

Changing a Client's Boot Release

In the case where a client is attached to more than one release, the Set operation allows you to change the default boot path. Adding a client to a new release with the 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 Defaults menu 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

When you add an OS client, **sysadm** requires that you set several parameters. As it often does, **sysadm** offers default values for the parameters. If the defaults are not what you want, however, you may supply instead your own defaults in what is called a defaults set. Use this operation to create such a defaults set.

Once you have created the defaults set, use the Modify operation to set the values for the defaults set. Use the Set operation (in the OS Client operation) to set the current defaults set.

Removing a Defaults Set

Use the Remove operation to delete an OS client defaults set created with the Create operation. Deleting a defaults set has no effect on clients added using the default set.

Modifying a Defaults Set

Use the Modify operation 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.

Listing Defaults Sets

Use the List operation to display the parameters for an OS client default set. See the section on the OS Client menu's Add operation for a discussion of each parameter.

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.

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.

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 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.

Adding an X Terminal Client

Select the 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.

Before performing the Add operation, add the X terminal client's Internet and Ethernet addresses to your TCP/IP databases. See the Networking menu. The X terminal displays its Ethernet address when you turn on power. Get the Internet address from your network administrator.

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.

Deleting an X Terminal Client

The Delete operation deletes 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 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

Use this operation 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 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

Chapter 7
Managing Disks

This chapter covers the tasks involved in managing your disks. These tasks include:

- Formatting and creating system areas on physical disks
- Copying physical disks
- Creating and maintaining striped and normal (nonstriped) logical disks
- Creating and maintaining software-mirrored disks
- Creating and maintaining cached disks
- Setting up and maintaining failover disk
- Creating, deleting, and changing the size of file systems
- Verifying integrity of and repairing file systems

Before you proceed, learn these terms:

Physical disk
A physical disk is the disk drive hardware. The SCM (the AViiON hardware's System Control Monitor) and the DG/UX operating system refer to physical disks with DG/UX device names such as **sd(insc(),0)** and **cimd(1,1)**. For more information on device names, see *Customizing the DG/UX™ System.*

Logical disk
A logical disk is a named set of one or more physical disk partitions, portions of disk space of specific sizes. The logical disk is the unit on which you typically create file systems. Therefore, file system operations (such as checking and repairing file system integrity, mounting file systems, and backing up file systems) operate on logical disk units rather than on physical disk units. You may divide a single physical disk into several logical disks. Conversely, you may spread a logical disk across several physical disks. A typical DG/UX system uses logical disks called **root**, **usr**, **swap**, and any others created on a site-specific basis. Of the DG/UX system logical disks (**root**, **usr**, and **swap**), **swap** is unique because it does not contain a file system.

File system
A file system is a software-formatted portion of disk space located on a logical disk. The file system contains the internal data structures that the operating system requires to keep track of files and directories on the logical disk. Typically, you build a file system on every logical disk that you create except on logical disks used as swap area (for demand paging) and any logical disks to be used for direct access by applications such as database managers.

Features for Improving Disk and File System Service

The DG/UX system offers a number of features that allow you to get better service from your disk file systems. These features can improve performance and data availability as well as provide compatibility with non-DG/UX file systems.

The following sections summarize these features.

Memory File Systems

For applications that would benefit from very fast access to relatively small databases, you can create memory file systems. A memory file system is a portion of your computer's physical memory, formatted as a DG/UX file system and mounted. You can access it the same as any file system. For more information on memory file systems, see "Creating a File System" in Chapter 8.

Software Disk Mirroring

You can improve the reliability and availability of your Data General AViiON system by creating software disk mirrors. Disk mirroring involves maintaining redundant logical disks where all are "mirror images" of each other; they all contain the same data. The system manages access to the disks in a manner that is transparent to users.

Disk mirrors provide higher data availability by allowing your system to continue service to users even when disk errors occur. Disk mirrors also protect data integrity by maintaining redundant images of the same data. In limited cases, mirrors can improve disk I/O throughput and thus improve overall system performance.

There are two ways to configure disk mirrors: through the hardware or through the operating system software. To configure hardware mirrors, you need a disk-array subsystem. This manual does not cover hardware disk mirrors. See the disk–array subsystem documentation for more information.

To create software disk mirrors, use the Disk Mirror Management Menu of **diskman**. See the section on mirroring in this chapter for more information on software disk mirrors.

Software Data Striping

Applications that perform a lot of random I/O (reads as well as writes) and applications that perform a lot of sequential reads can benefit from data striping. Data striping involves storing sequential file elements on alternating physical disks so that sequential disk accesses may be interleaved, taking advantage of the disk hardware's read-ahead feature to speed disk I/O.

From the administrator's point of view, a striped logical disk is as easy as any other logical disk to create and maintain because the system handles the data striping itself. The section on creating logical disks tells how to create software-striped disks and describes the kinds of applications that can benefit from data striping.

For information on hardware data striping, see the disk–array documentation.

Fast Recovery File Systems

To reduce the amount of time that the system will require to recover a file system after a failure, mount the file system with **fsck** logging turned on. With **fsck** logging, the system logs file system modifications to reduce the amount of time that **fsck** requires to verify the integrity of the file system. This feature is desirable for systems where rapid recovery and high availability are crucial. For more information on fast recovery file systems, see the section on **fsck** in Chapter 8.

Write Verification

In applications where data integrity is vital, you can benefit from write verification. By turning write verification on for a disk, you can be sure that data written to the disk is readable.

You can enable write verification only for SCSI disks that support the feature. See your disk hardware documentation.

Normally, disk hardware does not verify that data just written to disk is readable. This behavior makes your data vulnerable to flaws in the storage medium because the disk has no way of detecting when it has just written to a flawed block on the disk, for example. With write verification turned on for a disk, however, the disk hardware verifies every write operation by reading the written data off of the disk and comparing it to the data as originally received. If the data read from disk does not match the data originally received in the write request, the hardware returns an error to the system. Thus, write verification ensures the integrity of your data.

The tradeoff with write verification is in performance. The additional verification overhead in the hardware can have a significant impact on the performance of write-intensive applications. You should experiment with your applications to see how write verification affects performance. Write verification has no effect on read operations.

To turn write verification on for a disk, use the **dkctl**(1M) command with the **wchk** option. First, you need to get the path for the physical disk drive, for example, **/dev/pdsk/0**. Look in the **/etc/devlinktab** file for a list of drives on your system. By comparing the disk's device name with the long names in **devlinktab**, you can find the correct entry and note the short name. Use the short name in the **dkctl** command line. See *Customizing the DG/UX™ System* for more information on device names.

For example, the following command line turns on write verification for disk **/dev/pdsk/0**:

```
# dkctl /dev/pdsk/0 wchk ↵
```

This command line enables write verification for the disk every time you boot the system. To enable write verification only until the next system boot, include the **–t** (temporary) option:

```
# dkctl –t /dev/pdsk/0 wchk ↵
```

To turn write verification off for the disk, issue this command:

```
# dkctl /dev/pdsk/0 –wchk ↵
```

See the **dkctl**(1M) manual page for more information.

Disk Caching

Disk caching associates two data storage devices, typically a small, fast one with a large, slow one, so that an application uses the fast device for read and write operations while the operating system duplicates these operations on the larger device. The purpose of the configuration is to accelerate file system access for I/O-intensive applications without risking data integrity.

Device Sharing and Disk Failover

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

CD-ROM, Diskette, and Magneto-optical Disk Drive Support

The DG/UX system supports various SCSI disk drives including CD-ROM, diskette, and magneto-optical disk drives. For more information on these drives, see Chapter 15.

Non-DG/UX File System Support

The DG/UX system supports MS-DOS, High Sierra, and ISO 9660 file systems. MS-DOS file system support is particularly useful for sites with diskette devices, while High Sierra and ISO 9660 support is useful in environments with CD-ROM disk drives. Chapter 15 covers diskette and CD-ROM drives and several other types of SCSI drives. Chapter 8 covers file system operations. These chapters also include instructions for configuring your kernel for non-DG/UX file systems and for creating and mounting MS-DOS file systems.

Choosing the Right Tool for the Job

You perform disk management tasks using either **sysadm**(1M) or **diskman**(1M). In **sysadm**, the operations are under Device –> Disk. For more information on **sysadm**, see the manual page or simply invoke the utility and select the Help menu.

The **diskman** utility actually appears on your system in two forms that have virtually identical user interfaces. These two forms are:

Stand-among **diskman**

This version, **/usr/sbin/diskman**, is for use while the DG/UX system is booted and running. You can invoke this version only if you are the superuser. As an alternative, you may prefer to invoke **sysadm** instead because of its friendlier user interface.

Stand-alone **diskman**

This version, **/usr/stand/diskman**, is for use while the DG/UX system is not running. You boot this version to perform operations that you cannot perform while running the installed version of the DG/UX software. For example, if you wanted to expand the size of the **usr** logical disk, you would have to take the system down and boot stand-alone **diskman**, for example:

SCM> **b sd(cisc(),0)usr:/stand/diskman** ⤶

You can boot stand-alone **diskman** from disk only if the logical disk containing the **/usr** file system is a single-piece logical disk.

Although **diskman**'s interface is similar to **sysadm**'s, it offers only an ASCII terminal interface and no OSF/Motif interface. Table 7–1 summarizes the commands you can use with **diskman**.

Table 7–1 diskman(1M) Commands

User Input	Description
^ <New Line>	Return to previous menu or quit this operation.
? <New Line>	Print HELP message, then display menu.
number <New Line>	Choose menu item by entering a number.
number? <New Line>	Give information on the item number specified.
q <New Line>	Exit from **diskman**. You may need to enter **y** next.
<New Line>	Same as entering the default response.

If **diskman** receives invalid input, it displays a message indicating what features valid input should have, and then it displays the menu or question again. If an operation fails, **diskman** prints an error message prefaced with:

```
**** Error:
```

If you type ? after receiving an error message, **diskman** displays a HELP screen.

The tasks covered in this chapter indicate which utility—**diskman**, **sysadm**, or both—you may use to perform a given operation.

Making a New Disk Accessible

This section applies to normal hard disks. For devices such as CD-ROM, magneto-optical, and diskette drives, see Chapter 15.

To make a physical disk accessible to users on your system, follow these steps:

1. Format the physical disk and create system areas on it. Use the operations in **diskman**'s Physical Disk Management –> Format a Physical Disk menu or **sysadm**'s Device –> Disk –> Physical menu.

2. Register the physical disk. Use the Register operation in **diskman**'s Physical Disk Management –> Register, Deregister, or List Registered Physical Disks menu or **sysadm**'s Device –> Disk –> Physical menu.
3. Create one or more logical disks on the physical disk. Use the Create a Logical Disk operation in **diskman**'s Logical Disk Management menu, or use **sysadm**'s Device –> Disk –> Logical –> Create operation.
4. Create a file system on each logical disk. Use the Make operation in **diskman**'s File System Management menu, or use the Create operation in **sysadm**'s File System –> Local Filesys menu.
5. Add the file systems with the **sysadm** operation File System –> Local Filesys –> Add. You cannot perform this operation using **diskman.**
6. Make the file systems accessible to users by mounting them with the **sysadm** operation File System –> Local Filesys –> Mount. You cannot perform this operation using **diskman.**

This chapter covers steps 1 through 4. See Chapter 8 for steps 5 and 6.

Managing Physical Disks

Managing physical disks can involve a number of tasks, such as performing surface analysis, creating system areas, registering, deregistering, adding labels, and so on.

Configuring a Physical Disk

To make a physical disk accessible for any purpose, configure it using the **sysadm** operation Device –> Disk –> Physical –> Configure. This operation requires that you enter the device name of the physical disk, for example, **sd(dgsc(),1).**

By default, at boot the system configures all disks that have entries in the system file used to build the kernel. By default, the system file contains entries for any physical disks installed at standard locations when the system file was built. Configuring a physical disk puts an entry for it in **/dev/pdsk** and **/dev/rpdsk.**

Deconfiguring a Physical Disk

To make a disk inaccessible, deconfigure it using the **sysadm** operation Device –> Disk –> Physical –> Deconfigure. The operation requires that you select the desired physical disk's device name, for example, **sd(cisc(),0)**, from a list of configured disks.

If the disk is registered, you must deregister it before deconfiguring it. If the physical disk is mounted as a file system (as is typically the case with diskettes), you must unmount it before deconfiguring it. Deconfiguring a disk removes its entries from **/dev/pdsk** and **/dev/rpdsk.**

Soft Formatting a Physical Disk

Formatting a physical disk involves installing or creating three components on the disk: the disk label, which describes the disk medium to the DG/UX system kernel; the system areas, which are portions of disk space containing tables for bad blocks, logical disk pieces, and so on; and the bootstrap code, necessary if you plan to boot any image on the disk. You install or create these three components using separate operations offered in both **sysadm** and **diskman**. The following sections discuss these operations.

Installing a Physical Disk's Label

Disk labels contain the disk geometry (such as tracks per cylinder, bytes per sector, and so on), information that the system requires to write to the disk, read from it, and keep track of damaged disk blocks. To assign a label to a disk, use the **sysadm** operation Device -> Disk -> Physical -> Soft Format -> Install Label or the **diskman** operation Install a Disk Label on a Physical Disk, located in the Physical Disk Formatting Menu. You can assign a disk label only if the disk is deregistered.

The system does not use the labels on SCSI disks, but you may have to label them anyway when you create system areas on the disk. If you display the label of a SCSI disk, any or all of the label fields may be zero (0).

If the disk is a SCSI disk, the operation installs the generic SCSI disk label, which is the same for all SCSI disks, and terminates. If the disk is not a SCSI disk, it allows you to choose one from a set of predefined labels. If you select none of the predefined labels, you will need to enter 22 parameters for your disk when prompted. Your disk's hardware documentation should contain the relevant information. The prompts you need to answer are:

```
Enter the total cylinders per drive:
Enter the OS visible cylinders per drive:
Enter the tracks per cylinder:
Enter the sectors per track:
Enter the bytes per logical sector:
Enter the bytes per unformatted sector:
Enter the MFG defect information start sector:
Enter the Number of relocation areas:
Enter the Sectors per relocation area:
Enter the Next relocation sector:
Enter the interleave:
Enter the head skew:
Enter the cylinder skew:
Enter the head group skew:
Enter the spares per track:
Enter the bytes per data preamble:
Enter the bytes per id preamble:
Enter the base head for volume:
Enter the bytes in gap 1:
Enter the bytes in gap 2:
Do you want to have the disk contain a final short sector?
Does the drive use extended addressing on the disk?
```

To display a disk's label, select the **diskman** operation Display a Physical Disk's Label or the **sysadm** operation Device -> Disk -> Physical -> List.

Creating DG/UX System Areas

If you want to create a system area on a physical disk, select the **sysadm** operation Device -> Disk -> Physical -> Soft Format -> Create System Areas or the **diskman** operation Create DG/UX System Areas on a Physical Disk, located in the Physical Disk Formatting Menu.

CAUTION: *This operation destroys all data on the physical disk. Any logical disk having pieces on this physical disk will become completely inaccessible.*

System areas are portions of disk space allocated for both user and system data. System areas for system data contain various kinds of information required to manage logical disks and track bad blocks on the physical disk. System areas for user data are logical disks or logical disk pieces. You cannot create logical disks on a physical disk without first creating system areas on the physical disk.

Among the system areas that the operation creates are the Primary System Area (PSA) and a Logical Disk Piece Table that describes the logical disks that are on the physical disk. Another system area is the Bad Block Table which the system uses to remap bad blocks on the disk medium. When you create system areas on a disk, the operation calculates the size of the Bad Block Table (also called the bad block remap area) based on the overall size of the disk. This default size is generally sufficient.

When preparing a diskette, there is no benefit in creating system areas on the diskette. Diskettes are so small that the system areas would use up too much space. Typically, you create a single file system on the entire diskette. For more information on preparing diskettes, see Chapter 15.

Installing a Bootstrap

A physical disk must have a current bootstrap if you are to boot from the disk. Install the current DG/UX system bootstrap with the **sysadm** operation Device -> Disk -> Physical -> Soft Format -> Install Bootstrap or with the **diskman** operation Install a Bootstrap on a Physical Disk, located in the Physical Disk Formatting Menu.

If you are adding a new release of the DG/UX system, you need to reinstall the bootstraps on the disks containing the **root** and **usr** logical disks. You should install bootstraps on any disk from which you intend to boot, whether booting the kernel (**/dgux**), stand-alone **diskman** (**/usr/stand/diskman**), or any other bootable image.

Performing Surface Analysis

To verify the surface of the disk medium and have the DG/UX system map any bad blocks that it finds, select the **diskman** operation Perform Surface Analysis on the Physical Disk, located in the Physical Disk Formatting Menu. There is no **sysadm** operation to perform surface analysis; however, you may use the **admpdisk** command with the **verify** option to perform the same function. See the **admpdisk**(1M) manual page for more information.

CAUTION: *This operation destroys all data on the disk.*

If the disk is on a controller that performs hardware bad block mapping, the operation informs you and offers to cancel the operation if you wish. The operation allows you to choose all or any of several analysis patterns.

The operation reports estimated time-to-completion as it proceeds. When it finishes, it reports the number of bad blocks found and mapped.

Data General disks do not require surface analysis, but you may perform surface analysis if you wish. Running all test patterns takes about 20 minutes per 100 MB, depending on your physical disk model and CPU.

Performing All Soft Formatting Steps

To perform all of the soft formatting steps (install disk label, create system areas, install bootstrap), select the **sysadm** operation Device –> Disk –> Physical –> Soft Format –> All Soft Formatting Steps or the **diskman** operation All Formatting Steps, located in the Physical Disk Formatting Menu. The **diskman** operation also allows you to perform surface analysis although the **sysadm** operation does not.

Registering a Physical Disk

To make a disk's logical disks accessible on the system, register the disk either with the **diskman** Register operation in the Disk Registration Menu or with the **sysadm** operation Device –> Disk –> Physical –> Register. You need to know the physical disk's device name, for example, **sd(dgsc(),1)**, in order to register it.

You can register a physical disk only if it contains system areas. System areas are used by the system to track logical disks and bad blocks. You create system areas with Device –> Disk –> Physical –> Soft Format –> Create System Areas. Diskettes and CD-ROM disks that contain High Sierra or ISO 9660 file systems do not contain system areas and do not require registration. For more information on diskette and CD-ROM disk drives, see Chapter 15.

Registering a physical disk makes the physical disk's logical disks known to the system. You cannot access a logical disk unless you register the disk drive containing it.

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

Deregistering a Physical Disk

To make a disk's logical disks inaccessible, deregister the disk either with the **diskman** Deregister operation in the Disk Registration Menu or with the **sysadm** operation Device –> Disk –> Physical –> Deregister. You need to know the physical disk's device name, for example, **sd(dgsc(),1)**, in order to deregister it.

If a file system on the physical disk is still mounted, or if a logical disk is open for any other purpose, deregistration fails.

You should deregister a disk before removing it from the system. Before removing a CD-ROM or magneto-optical disk from the disk drive, unmount the file systems and deregister the physical disk (if registered). If you have deregistered a CD disk, optical disk, or diskette drive, you may remove the disk from the drive without turning off power to the computer or drive.

If you have deregistered the disk in order to disconnect the disk drive from the system, make sure you shut down the system and turn off power to the computer and disk drive before removing it.

Copying a Physical Disk

To copy the logical disks of one physical disk to another physical disk, select the **sysadm** operation Device –> Disk –> Physical –> Copy or the **diskman** operation Copy a Physical Disk, located in the Physical Disk Management Menu. You can use this operation to copy between disks of different types and sizes.

Why and When to Copy
There are several cases where you may want to copy a physical disk:

- When you replace an old disk with a new one, you can copy the contents of the old one to the new one.
- When installing a number of systems that need the entire operating system loaded from tape, you can save time by loading only the first disk from tape and then copying the other disks from it.

  To use the copy facility for this purpose, you may need to know how to jumper the destination disk for a different address or SCSI ID. This is particularly true if you intend to use the copy facility to load one workstation's SCSI disk from another workstation's SCSI disk. Typically, both SCSI disks come from the factory jumpered as **sd(insc(),0)**. Before connecting them both to the same system, you need to change the jumpers on one of them to put it at another SCSI ID, for example, **sd(insc(),1)**.

  If you use the copy facility to make copies of the DG/UX system logical disks (**root** and **usr**), perform the copy before setting up the DG/UX system. If you perform the copy after setting up, you will duplicate system-specific data (Internet addresses, system names, and so on) from the source disk.
- When you need to remove a disk from your system, use the Copy operation to merge the contents of the disk onto another disk having the required available space.

There may also be other scenarios where you find the physical disk copy operation useful.

How to Copy
Before attempting the Copy operation, make sure both disks already have system areas. To create system areas on a disk, use either the **sysadm** operation Device -> Disk -> Physical -> Soft Format -> Create System Areas or the equivalent **diskman** operation in the Disk Formatting Menu.

The operation does not necessarily copy the logical disks to the exact same locations on the destination disk that they occupied on the source disk. The operation attempts to fit the logical disks or logical disk pieces as efficiently as possible into the available space on the destination disk. If possible, the operation coalesces any free space left on the destination disk into a single area at the end of the disk.

If the operation encounters errors, it terminates and removes from the destination disk any logical disks or logical disk pieces that it copied during the operation.

Error messages that appear during the copying process, such as errors indicating insufficient disk space on the destination disk, may include the name of the source or destination logical disks.

During the copy process, the operation gives each destination logical disk a temporary name derived from the name of the corresponding source logical disk. The temporary name is the same as the source logical disk name except that the first character is incremented. For example, when copying existing logical disk **sales**, the operation creates destination logical disk **tales**. If a logical disk named **tales** already existed, the operation would fail.

The reason the operation renames destination logical disks is because the DG/UX system cannot recognize multiple logical disks having the same name. The temporary names remain as long as the copy operation continues. When the operation has copied the logical disks, it deregisters the destination disk and renames its logical disks so that they now have the same names as the original source logical disks.

NOTE: After the copy operation is complete, you may not register both the source and the destination disks. After copying or merging, the two disks contain identically-named logical disks, and the DG/UX system will not permit you to register physical disks containing identically-named logical disks.

When the copy operation is complete, the source disk is registered and accessible, but the destination disk is not. You will not be able to register the destination disk if it contains fragments of logical disks (that is, some but not all of a logical disk's pieces).

The **sysadm** and **diskman** Copy operations are not exactly the same. The most important difference is that the **sysadm** version can perform only a merging copy, while the **diskman** version can perform either a merging copy or an over-writing copy. A merging copy duplicates the contents of the source disk on the destination disk. After the operation, the destination disk contains its original contents plus those of the source disk. An over-writing copy, on the other hand, involves removing all logical disks from the destination disk before copying the source disk's contents to it.

The sysadm Copy Operation
The **sysadm** Copy operation requires that you register both disks before performing the operation. The operation merges the contents of both disks onto the destination disk.

The diskman Copy Operation
The **diskman** Copy operation requires that you register the source disk and deregister the destination disk before beginning the operation. If the destination disk has no label, the **diskman** operation asks if you want to install a label. You do not have to install a label. If you respond with **yes**, the operation either displays the disk label choices or, if the disk is a SCSI disk, installs the generic SCSI disk label without further prompting. If you respond with **no**, the operation continues.

The **diskman** operation then displays the following caution message:

```
CAUTION:  this operation will DESTROY any data on physical disk pdisk.
```

where *pdisk* is the name of the destination physical disk.

The operation asks if you want to continue. If you want to perform an over-writing copy, that is, copy the contents of the source disk onto the destination disk without preserving the contents of the destination disk, answer **yes**.

CAUTION: Answering yes at this point in the operation will destroy all data on the destination disk.

If you want to merge the contents of the source disk onto the destination disk, preserving the original contents of the destination disk as well as the source disk, answer **no**. The operation then asks you to confirm that you want to merge the contents of the two disks onto the destination disk. Answer **yes** to continue with the merge; answer **no** to terminate the operation.

Listing Physical Disks

To display a variety of information about physical disks configured on your system, select the **sysadm** operation Device -> Disk -> Physical -> List. Unlike most List operations available in **sysadm**, this operation is restricted to superuser access.

The List operation displays the device name and label of the physical disk, whether or not the disk is registered, its size in 512-byte blocks, and if registered, the number of free blocks remaining on the disk. As an option for registered physical disks, you may also display information about system areas and logical disk pieces on the physical disk. This additional information includes each system area's starting address and size in 512-byte blocks and, if a logical disk piece, the piece number and the name of the logical disk to which it belongs. You may retrieve this same information with the **diskman** operation Display a Physical Disk's Layout.

Tracking Bad Blocks on a Physical Disk

Disk units occasionally develop flaws in the disk surface. Most disk hardware keeps track of these bad parts without depending on the operating system to do it for them; nevertheless, the DG/UX system offers its own bad block tracking mechanism in case the disk hardware fails to detect or remap a flawed disk block.

When the DG/UX system detects a flaw on a disk, it flags the block (a 512-byte portion of disk space) as bad and finds a good block to replace it. The operating system takes care of redirecting reads and writes intended for the bad block so that they go to the replacement block instead. A part of the disk called the *bad block remap area* contains good blocks reserved specifically for this purpose: to replace blocks that go bad elsewhere on the disk.

The system creates the bad block remap area when it creates system areas. When the system creates the bad block remap area, it offers a default remap area size that is based on the size of the disk. You may specify another size if you like, but the default size is generally sufficient.

Mapping Bad Blocks

To add bad blocks to a physical disk's bad block table, select the **sysadm** operation Device -> Disk -> Physical -> Bad Blocks -> Map or the **diskman** operation Add Bad Blocks to a Physical Disk's Bad Block Table, located in the Bad Block Management Menu.

Bad blocks are parts of the physical disk that may be unreliable for storage and retrieval of information. The operating system keeps track of bad blocks by listing them in its bad block table. There may be other bad blocks besides the ones listed in the bad block table. If you suspect that a particular block is unreliable or if diagnostics have shown a block to be unreliable, you can add that block to the bad block table using the operations named above.

To add a bad block to the bad block table, you need to know the device name of the physical disk and the physical address of the bad disk block. The disk should be registered before you invoke the operation. You do not need to know the address of the replacement block.

If you map a block that contains data, the system will not attempt to copy the data from the bad block to the replacement block. The contents of the replacement block are unknown until a write operation writes data to the block.

In the **diskman** operation, if the physical disk is not registered, the operation offers to register it before continuing. In the **diskman** operation, if any of the bad blocks are readable (not bad according to the system), you are informed about them and asked if you still want to add those blocks to the Bad Block Table.

If the bad block remap area does not have any more good blocks (which is highly unlikely), the operation returns an error message telling you that it cannot add the bad blocks to the table. When this happens, dump the disk contents to tape and recreate the system areas on the disk, being sure to specify a larger bad block remap area size. Then reload the disk from tape.

Unmapping Bad Blocks

To recover bad blocks from the bad block table, use the **sysadm** operation Device -> Disk -> Physical -> Bad Blocks -> Unmap or the **diskman** operation Recover Bad Blocks from a Physical Disk's Bad Block Table, which appears in the Bad Block Management Menu.

The operation recovers the remapped blocks that no longer need to be remapped. To remap a block means to designate another block to use in place of the bad or unreliable block. For example, block 100000 might be remapped to another block. After getting your disk serviced, however, you may want to remove block 100000 and all other formerly bad blocks from the bad block table.

Listing Bad Blocks

To display a physical disk's bad block table, select the **sysadm** operation Device -> Disk -> Physical -> Bad Blocks -> List or the **diskman** operation Display a Physical Disk's Bad Block Table, which appears in the Bad Block Management Menu.

After you have formatted a disk, use this operation to display bad blocks (if any). If you supply the disk drive name, the operation lists the addresses (in decimal form) where bad blocks are located.

Managing Logical Disks

Logical disks are formatted pieces or sections of physical disks. You might think of a logical disk as a virtual disk because it can have pieces on more than one physical disk, but it functions as a whole.

A logical disk name may be no more than 31 characters in length. Legal characters are **a-z**, **A-Z**, **0-9**, **-** (hyphen), **_** (underscore), and **.** (period).

You may name your logical disks according to a function or classification that fits your application, such as a logical disk named **tax_86** that contains tax records for 1986.

Creating a Logical Disk

You can create logical disks only on a soft-formatted physical disk (that is, one containing system areas). To perform soft formatting, use the **sysadm** operation Device -> Disk -> Physical

-> Soft Format -> Create System Areas or the equivalent **diskman** operation in the Physical Disk Formatting Menu.

When you have soft-formatted your physical disk, you are ready to create one or more logical disks. Use the **sysadm** operation Device -> Disk -> Logical -> Create. In **diskman**, create a logical disk by selecting the Create operation in the Logical Disk Management Menu.

If you have applications that perform a lot of random reads and writes in a file, or applications that perform a lot of sequential reads, you may find that you can improve disk I/O performance by striping data on logical disks. If you intend to stripe data, you need to create the logical disk specifically for this purpose. A discussion of data striping appears in the next section.

You may create a logical disk in multiple pieces. The advantage of a multi-piece logical disk is that you can distribute it across several physical disks, thus improving I/O performance. The ability to create a multi-piece logical disk is also helpful in cases where a physical disk does not have the desired amount of space in a single contiguous area of the disk. In such cases, you can join any portions of any free areas on the physical disk into a single logical disk. You can create a logical disk having up to 32 pieces and use it just as if it were a single-piece logical disk. The disadvantage to having multi-piece logical disks on separate physical disks is that if a hardware failure makes any piece of the logical disk inaccessible, the entire logical disk also becomes inaccessible, even pieces residing on functioning physical disks.

The Create operation prompts you to create the first piece of the logical disk first. When finished, it allows you to create additional pieces if desired.

For each piece, you select a starting address and a length. The operation offers to display the physical disk's current space allocation if desired, and it allows you to choose another physical disk if desired. You may create the logical disk pieces on different physical disks if desired.

Choosing a Starting Address

For each piece, you need to decide on a physical disk starting address. Physical disk addresses refer to the 512-byte blocks making up the disk. They start at 0 and continue throughout the disk. The beginning of a disk is allocated for system areas (including the bad block table), so the first available address when creating a logical disk will be several hundred blocks into the physical disk.

Deciding on Size

You also need to specify a length for each logical disk piece. Like the starting address, the length is in 512-byte blocks. If you are creating a striped disk, you specify length only for the first piece of the disk. See the following section for more information on striped disks.

Unless you are creating a logical disk for which there is an assigned default size (such as **root**, **usr**, or **swap**), the default size for the logical disk is the maximum possible size, given the current layout of the physical disk.

When you select the size for a logical disk, it is important to keep in mind that all of the space in the logical disk will not be available for storing files. File system overhead, in the form of internal data structures, will consume some of the logical disk space, thus making it unavailable to you. A DG/UX file system by default also has a reserved free space buffer (10% by default) for which you must plan. Assuming that you create a DG/UX file system with the default characteristics, you should create the logical disk 17% larger than the amount of data that you intend to store in the file system.

For example, if you need a DG/UX file system to hold 100 MB of data, create a logical disk 117 MB in size. When the file system contains no files, it is around 4% full. After loading it with 100 MB of data, it is 90% full. When a file system is 90% full, only the superuser can extend or create files or directories in the file system.

If you decide at any time that you have created a logical disk too large or too small, you can change its size with **sysadm**'s Expand and Shrink operations, which appear in both the Device -> Disk -> Logical menu and the File System -> Local Filesys menu.

If you specify more pieces, the operation repeats the starting address and size queries. If you intend to boot from the logical disk, do not add more pieces. You cannot boot from a multi-piece logical disk.

Creating a File System

After creating a logical disk with **sysadm**, you create a file system on the logical disk with the operation File System -> Local Filesys -> Create. If you used **diskman** to create the logical disk, it will offer to create the file system as soon as the logical disk creation operation succeeds. See Chapter 8 for complete information on creating a file system.

Whether you use **sysadm** or **diskman** to create the file system, you may specify **mkfs** options or arguments for the create operation. The **mkfs** utility is what the system uses to create a file system. See the **mkfs**(1M) manual page for more information.

Striping a Logical Disk

Depending on the nature of your applications, you may find that data striping improves disk I/O performance. You can implement data striping through the hardware (if you have a disk–array) or through the software. This manual covers only software data striping. For information on hardware data striping, see your disk-array documentation.

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

Briefly, the implementation of striping consists of placing consecutive file elements in the file system so that they alternate from one piece of the logical disk to the next. For example, in the case of a striped three-part logical disk, each part must be the same size and each must reside on a different physical disk. The system will place the first data element in the first logical disk piece, the second data element in the second logical disk piece, and the third data element in the third piece. The data elements alternate this way so that consecutive data elements are stored on alternating disks. The performance advantage results not only because you have distributed the I/O load across three disks, but also because you are using the hardware's read-ahead implementation to get the next element on that disk, even before you have explicitly requested it.

Before continuing, you need to consider whether or not striping can benefit your application's performance. First, striping is only possible if you have more than one physical disk. Second, striping only helps applications that perform a lot of random reads and writes or a lot of sequential reads. Data striping does not help applications that perform a lot of sequential writes.

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

The logical disk that you intend to create for striping must conform to these guidelines:

- The logical disk must consist of multiple pieces.
- Each piece must reside on a different physical disk.
- The pieces must all be the same size, and the size must be a multiple of the stripe size (16 blocks by default).
- You may not stripe an existing file system or logical disk. You may only stripe a logical disk when you create it.
- You may not stripe the root or **/usr** file systems.

The **sysadm** Create operation prompts you for the name of the logical disk first, then prompts whether you want to stripe the logical disk. If you choose to stripe it, the operation prompts for the logical disk piece size only for the first logical disk piece. The operation uses this value as the size for the other logical disk pieces. If you attempt to create a piece on a physical disk that already contains a piece of this striped logical disk, the operation returns an error.

The **diskman** Create operation prompts you for the starting address and size of each logical disk piece. If the pieces are all the same size and are each located on different physical disks, the operation asks if you want to stripe the logical disk. If your logical disk does not satisfy these conditions, the Create operation does not allow you to stripe it.

When creating a striped logical disk, the operation asks what stripe size to use. The stripe size must be no smaller than the data element size, and it must be a multiple of the data element size. The default data element size is 16 blocks, so the stripe size may be one of 16, 32, 48, and so on. Remember that the logical disk size must be a multiple of the stripe size.

Once you have created the striped logical disk and file system, you may use the file system just like any other file system.

To list information about a striped logical disk, or to see if a logical disk is striped, use the **sysadm** operation Device –> Disk –> Logical –> List.

Removing a Logical Disk

To remove a logical disk, select the **sysadm** operation Device –> Disk –> Logical –> Remove or the **diskman** Delete operation from the Logical Disk Management Menu. This operation removes all pieces of the logical disk.

All pieces must be intact and accessible for the operation to succeed. This means that any physical disks containing pieces of the logical disk must be registered. If a hardware failure or other problem prevents you from accessing all pieces of a logical disk that you want to remove, see "Removing a Piece of a Damaged Logical Disk."

You cannot remove a software-mirrored disk or a logical disk currently in use as a mirror image. To remove a mirror or an image, first break the mirror into its component logical disks using Device –> Disk –> Software Mirrored –> Break. Then remove the logical disks with the Remove operation.

Displaying Information About a Logical Disk

To display information about a logical disk, select the **sysadm** operation Device –> Disk –> Logical –> List. In **diskman** select either Display Information about a Logical Disk or Display Information about a Logical Disk Piece, both located in the Logical Disk Management Menu.

The **sysadm** List operation allows you choose among the individual logical disks, all logical disks, or all disks that are currently usable. The operation displays the following information for each logical disk:

`Logical Disk`	The name of the logical disk.
`Size`	The total number of 512-byte blocks making up the logical disk.
`Stripe`	Whether or not the logical disk is striped. The section on creating logical disks discusses striping.
`Usable`	Whether or not any of the logical disk's pieces is on a physical disk that is inaccessible or unregistered.
`In Mirror`	The mirror to which this logical disk belongs, if any. A later section discusses mirrors.

Additionally, you may choose to display information about each logical disk piece. For each piece, the **sysadm** List operation displays the following information:

`Piece`	The number of the piece relative to all the pieces making up this logical disk. Piece numbers reflect the order in which you created the pieces; from the system's point of view, piece numbers are arbitrary identifiers. The system attempts to balance the data storage and I/O loads evenly among all pieces of a logical disk, regardless of their piece number.
`Address`	The address on the physical disk where the logical disk piece begins.
`Size`	The size of this logical disk piece.
`Physical Disk`	The physical disk where this logical disk piece resides.

When displaying information about a logical disk or disk piece, **diskman** displays the following information:

`Piece`	The number of this piece relative to the other pieces making up the logical disk.
`Physical Disk`	The physical disk where this logical disk piece resides.
`Starting Physical Disk Address`	The address on the physical disk where the logical disk piece begins.

`Size in Blocks` The size of this logical disk piece.

Copying All or Part of a Logical Disk

To copy all or part of a logical disk, select the **sysadm** operation Device -> Disk -> Logical -> Copy. In **diskman**, you select either Copy a Logical Disk or Copy a Logical Disk Piece, both located in the Logical Disk Management Menu.

If copying an entire logical disk, the source and destination logical disks must be the same size. They physical disks on which they reside must be registered, and the logical disks must have different names.

If copying only a logical disk piece, the same rules apply as above: the pieces must be the same size, the physical disks containing the logical disks must be registered, and the logical disks must have different names.

You cannot copy to or from a logical disk whose file system is currently mounted or which is otherwise in use.

CAUTION: *This operation destroys any data on the destination logical disk or logical disk piece.*

Removing a Piece of a Damaged Logical Disk

You can remove a piece of a damaged logical disk in **diskman** by selecting Delete a Piece of a Damaged Logical Disk from the Logical Disk Management Menu. Whenever one piece of a logical disk becomes inaccessible, all other pieces become inaccessible, and thus the entire logical disk is inaccessible. To re-use physical disk space, you must delete all logical disk pieces.

CAUTION: *Use this procedure only for deleting pieces associated with damaged physical disks. If you try to delete a piece of a usable logical disk, you will make that entire logical disk inaccessible. If you wish to delete an entire usable logical disk, see "Removing a Logical Disk," earlier in the chapter.*

Deleting only a portion of a damaged logical disk allows you to recover physical disk space so that you can use it for other logical disks. This procedure is useful when a failure damages a physical disk containing a piece of a logical disk.

For example, consider a two-piece logical disk named **trimble** where piece 1 is on physical disk **sd(dgsc(),0)**, and piece 2 is on physical disk **sd(ncsc(),1)**. If **sd(dgsc(),0)** bursts into flame and is lost, you can no longer access any part of logical disk **trimble**. If you want to use the space on **sd(dgsc(),1)** that was piece 2 of **trimble**, you must first recover the space by deleting the piece using **diskman**.

Managing Software-Mirrored Disks

You can improve the reliability and availability of your Data General AViiON system by creating disk mirrors. You can set up disk mirroring through the hardware (if you have a disk–array), through the software, or both. This manual covers only software disk mirroring. For information on hardware disk mirroring, see your disk-array documentation.

Software disk mirroring involves setting up redundant disks that are all *mirror images* of each other: they all contain the same data. The system manages access to the disks in a manner that is transparent to users. For the administrator, disk mirrors are easy to maintain. Disk mirrors provide three benefits:

Data availability
Because the disks making up the mirror all contain the same data, a single disk failure no longer interrupts access to the data. As long as one disk in the mirror remains functional, users experience no interruption in service.

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

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

While a single running application will not exhibit increased performance, the system overall will exhibit more improved performance. This benefit does not occur in environments where only one running application reads the mirror at a time, nor does it occur in environments where the images do not reside on different physical disks.

You do not need any special hardware or software, other than the DG/UX system, to take advantage of the benefits of software mirroring; however, mirroring provides the most benefits on systems with multiple physical disks.

Understanding the Concepts

A software mirror consists of two or three logical disks, called images, that are the same size. Once associated with each other as a mirror, the logical disks are no longer visible under the **/dev** directory or available on your system as individual logical disks. Instead, the mirror appears as a logical disk on the system, accessible the same as a normal logical disk. You can perform any operation on the mirror that you can perform on a normal logical disk.

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

If a read operation fails on one image, the system satisfies the read request by reading from another image instead. If a bad block caused the original failure, the system attempts to repair it by remapping the bad block and updating it with the correct data from a good block on another image. If this repair operation succeeds, the image remains an active member of the mirror. If the repair operation fails, however, the system responds to the failure as it does to a failed write operation, described below.

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

When an image becomes corrupt, the system reports the event by logging a message using the **syslog** error logging facility. The message is from the **kern** facility and is at level **notice**. By default, **kern.notice** messages go to the system console and to the file **/usr/adm/messages**. To change how the system handles notice messages, see "Logging System Errors" in Chapter 3 and the manual page for **syslog.conf**(5).

Once an image becomes corrupt, the system considers it to be *out of sync* with the other images in the mirror. Being out of sync means that the data in the image is no longer consistent with the data in the other images: the images are no longer identical. After you remedy the cause of the failure and restore the image to service in the mirror, the system then needs to synchronize the image before considering it no longer corrupt. To synchronize an image, the system selects the most recently updated good image to act as a master image. The system then copies the master image onto the out-of-sync image.

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

At the beginning and end of a synchronization session, the system logs a message using the **syslog** error logging facility. The message is from the **kern** facility and is level **notice**. If synchronization fails, the system logs a **kern.err** message, which by default goes to the system console and to **/usr/adm/messages**.

Booting from a Mirror
To boot an image residing on a mirrored disk, specify the name of one of the available images rather than the name of the mirror itself. For example, if you created a software-mirrored disk for your **root** logical disk, and called the images **root1** and **root2**, you would boot by specifying either of the images, as in the following boot command line:

```
SCM> b sd(dgsc(),1)root2:/dgux ⤶
```

Operations for Managing Mirrors
The operations for disk mirrors are available in **sysadm**'s Device -> Disk -> Software Mirror menu or in **diskman**'s Disk Mirror Management Menu. There are operations to create a disk mirror, break a disk mirror, modify a mirror, display disk mirror information, and synchronize mirror images.

Creating a Software-Mirrored Disk

To create a disk mirror or add an image to a disk mirror, select **sysadm**'s Device -> Disk -> Software Mirror -> Create operation or in **diskman**, the Build operation in the Disk Mirror Management Menu. Before you invoke the operation to create a mirror, you need to decide several things:

- How many images will the mirror include?
- On which physical drives and controllers will you put the images?
- How will you configure the logical disks for each image?

- How many images must be available before mounting the mirror?
- Will the system synchronize images (as needed) at every boot? How quickly should the system perform the synchronization?

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

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

What kind of logical disks should you create?
Once you have decided on the number and location of the images, you need to make the decisions that are pertinent any time you create a logical disk. The only requirement for mirroring is that all logical disks be the same size. The names of the logical disks and the number and placement of any logical disk pieces, however, are completely up to you.

How many images must be available?
You need to decide how many images must be available before you can mount the mirror and make it accessible to users. This question is just another way of asking how many corrupt images you will tolerate. For example, if your system has three images, you may decide that you want at least two images functioning (noncorrupt) before you can mount the mirror. On the other hand, you may feel safe enough functioning with only one available image.

Keep in mind that you may not be on hand to evaluate the state of the system after a system crash. After a power outage, for example, a Data General AViiON system reboots itself without operator intervention as soon as power returns. Depending on how you configured the system, it may come up all the way to run level three, where local users and OS clients may begin work and access file systems. The availability requirement you set will determine whether users can access a mirror that may have lost an image in the crash. If the data in the mirror is such that you do not want it in use if there is only one functioning image, you should set the availability requirement to two.

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

How should the system perform synchronization?
For every mirror you create, you need to define how synchronization should occur when necessary. First, you need to decide whether the system should synchronize the images in the mirror, if necessary, at every boot, or whether the system should activate the mirror without checking to see if synchronization is necessary.

Second, you decide how quickly synchronization should occur. You may choose fast synchronization, where the system updates the out-of-sync image as quickly as it can (many updates a second), or slow synchronization, where the system updates the out-of-sync disk at a rate of about one update per second. Fast synchronization may have a considerable impact on system disk I/O performance. Slow synchronization, on the other hand, has less impact on system disk I/O performance, but it increases the length of time that the image is not completely functional.

The steps for creating a disk mirror are:

1. Create two or three logical disks of the same size, preferably on different controllers and disks. These will be the images for the mirror.
2. Create a file system on one of the images.
3. Use **sysadm**'s Create operation to create the mirror.

   You may name the mirror anything you wish. There must not already be a mirror or logical disk with that name on the system.

   When prompted to add logical disks to the mirror, add the logical disk with the file system first, then add any other logical disks after it.
4. Start synchronization for the images that do not have file systems. Use **sysadm**'s Device -> Disk -> Software Mirror -> Synchronize operation. The master image for synchronization should be the logical disk that has the file system. You will have to perform the synchronization operation once for every image that you are synchronizing. This step copies the file system data from the master image to the slave images.
5. Mount the file system.

You may now use the mirror as a normal file system.

Once you add logical disks to a mirror, they are no longer accessible on your system as individual logical disks. Instead, the mirror exists as a single logical disk representing them. Any operations that you may perform on a logical disk, you may perform on the mirror. File system operations such as **dump2** and **restore** function on the mirror's file system the same way that they function on any other file system.

Breaking a Software-Mirrored Disk

To remove all images from a disk mirror, select the **sysadm** operation Device -> Disk -> Software Mirror -> Break, or the **diskman** operation Break a Disk Mirror, located in the Disk Mirror Management Menu. Breaking a mirror does not destroy the data on any of the images; it simply disassembles the mirror into its component logical disks. You cannot break a disk mirror while synchronization is in progress.

To remove only some images from the mirror, use the **sysadm** operation Device -> Disk -> Software Mirror -> Modify, described in the following section.

Before you can break a mirror, you must unmount the mirrored file system. Do not unmount the mirror if synchronization is in progress. Unmount a mirror with the **sysadm** operation File System –> Local Filesys –> Unmount.

Once you have broken a mirror, you can use its logical disks just as you would any logical disks.

Modifying a Software-Mirrored Disk

To add an image to or remove an image from an existing software-mirrored disk, select the **sysadm** operation Device –> Disk –> Software Mirror –> Modify.

You may modify a mirror while it is mounted as long as you do not attempt to remove the last image. If a mirror has only one image remaining, an attempt to remove the image will fail. You can remove the final image only by first unmounting the mirror's file system with File System –> Local Filesys –> Unmount. Removing the final image is the same as breaking the mirror.

Removing an image from a mirror does not change the availability requirement that you set for the mirror. If removing the image causes the number of available images to drop below the mirror's availability requirement, you will not be able to use the mirror the next time you boot the system or attempt to mount the mirror. To make the mirror accessible again, do either of two things:

- Add one or more images with the Modify operation, or
- Use the Break operation to remove all images from the mirror, then use Create to recreate the mirror with a lower availability requirement.

Any images removed from a mirror still exist on the system as independent logical disks. If the image you remove is still fully functional, you may use it like any other logical disk.

You may add any existing logical disk to a mirror as long as the logical disk is not currently in use.

Synchronizing a Software-Mirrored Disk's Images

Under normal circumstances, the system maintains the exact same data in all images in a mirror. Under some conditions, however, one image in a mirror may become inconsistent with the rest. When this happens, use the Synchronize operation to bring the inconsistent image up to date.

To synchronize an image, select the **sysadm** operation Device –> Disk –> Software Mirror –> Synchronize or the **diskman** Synchronize operation, located in the Mirror Management Menu.

Before performing the operation, you need to know the name of an up-to-date image in the mirror. You specify this image as the master (source) image, and you specify the new or out-of-date image as the slave (destination) image. The system then copies the entire master image block-by-block onto the slave image, overwriting it entirely.

CAUTION: *Be very careful when selecting the master and slave images for the synchronization operation. If you accidentally specify a good image as the slave, the operation will destroy all data on the image.*

You do not need to unmount a mirror to synchronize an image in it. While synchronization is in progress, the system continues to allow access to the mirror. You cannot break a mirror on which synchronization is occurring.

You may select how quickly the system performs the synchronization. If you select fast synchronization, the system performs the synchronization as fast as possible, updating the slave image many times per second. Fast synchronization may have a considerable impact on disk I/O performance, possibly inconveniencing users on the system. If you select slow synchronization, the system performs only one update every second, causing less impact on system disk I/O but extending the time required for synchronization.

At the beginning and end of a synchronization session, the system logs a message using the **syslog** error logging facility. The message is from the **kern** facility and is level **notice**. By default, **kern.notice** messages go to the system console and to the file **/usr/adm/messages**. If synchronization fails, the system logs a **kern.err** message, which by default goes to the system console and to **/usr/adm/messages**. To change how the system handles notice and error messages, see "Logging System Errors" in Chapter 3 and the manual page for **syslog.conf**(5).

Do not attempt to unmount a mirror while synchronization is in progress.

Displaying Software-Mirrored Disk Information

To list information about disk mirrors, select the **sysadm** operation Device -> Disk -> Software Mirror -> List or the **diskman** operation Display Disk Mirror Information, located in the Disk Mirror Management Menu.

The information displayed largely corresponds to the attributes you gave the mirror when you created it: the name, the number of images present, the number of images required, the autosync setting, and the size of the mirror (that is, the usable size, not the cumulative size of the component images).

The display also includes the current status of the mirror and its images. The current status information includes the sync status of any images: whether the image is synched or unsynched (that is, corrupt) and, if synchronization is in progress, the percentage of synchronization complete and the name of the image acting as the sync master.

Managing Cached Disks

Disk caching associates two data storage devices, typically a small, fast one with a large, slow one, so that an application uses the fast device for read and write operations while the operating system duplicates these operations on the larger device. The purpose of the configuration is to accelerate file system access for I/O-intensive applications without risking data integrity.

Currently, the only supported disk caching configuration is also the ideal configuration. It consists of a nonvolatile random access memory (NVRAM) or battery backed-up random access memory (BBURAM) board functioning as the fast device (the *front end*) while a physical disk functions as the slow device (the *back end*). Although the RAM board has a relatively small storage capacity, its superior I/O performance can boost the performance of I/O-intensive applications such as database management systems.

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

The disk functioning as the back end, meanwhile, provides greater storage capacity than a RAM device and has the added stability normally attributed to disk drives. The back end device can be a local disk or a remote one whose file system is mounted using ONC/NFS.

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

How Caching Works

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

As I/O access to the cached disk continues, the system allocates more buffers until the system reaches a *high-water mark.* The high-water mark, expressed as a percentage of the total number of buffers, determines how full the cache will become before the system begins reclaiming buffers and making them available again for allocation. The system continues to free buffers in this manner until it reaches a *low-water mark,* which is also expressed as a percentage of the total number of buffers.

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

To determine which buffers are the least frequently accessed, the system maintains an *access weight number* for each buffer. Each time an I/O request accesses the buffer, the system increments its access weight number. When the time comes to reclaim buffers, the system can then compare the access weight numbers of the buffers to see which are most frequently accessed (and should stay in the cache) and which are least frequently accessed (and may be freed).

When you shut down the system with the **shutdown** command, access to the cached disk stops just as for any other disk. The data in the cache, however, remains even if the cache contains dirty buffers whose data should be flushed to disk. The cache does not flush its dirty buffers to disk until you halt the DG/UX system using the **halt** command. When the cache has finished flushing its buffers to disk, it flags all buffers as clean.

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

When the DG/UX system attempts to register the cache device at boot time, the cache device checks the serial number of the CPU and compares it to the serial number of the CPU to which it was attached when the failure occurred. If the serial numbers match, registration succeeds and the cache device flushes its dirty buffers (if any) to disk as soon as the disk becomes available. If the CPU serial numbers do not match, the registration attempt aborts and an error message appears at the system console.

The purpose of aborting registration is to prevent the cache from flushing its buffers to the wrong disk in the event that you re-installed the cache device in another computer. By failing to register, the cache device gives you the option of explicitly forcing the flush yourself (with **diskman**'s Register Cached Device operation) or of destroying the data on the cache device and restoring it to a completely unallocated state (with the Create Cached Device operation).

Tuning a Cached Disk

The DG/UX disk caching feature provides several parameters you can tune to optimize cache efficiency. These parameters are:

Optimal I/O transfer size in bytes
: This value is the average expected size of a buffer. The system uses this number to determine the maximum number of buffers that the cache device can hold. The default is **8192** bytes (8K), which permits a 2 MB cache device to contain up to 255 buffers.

Read retention value
: This is the amount by which the system increments a buffer's access weight number when the buffer is accessed for reading. By default, the system increments a buffer's access weight number by **1** for a read request.

Write retention value
: This is the amount by which the system increments a buffer's access weight number when the buffer is accessed for writing. By default, the system increments a buffer's access weight number by **1** for a write request.

High water mark percentage
: This figure determines what percentage of the cache device's buffers must be allocated before the system will begin reclaiming buffers and making them available again for allocation. By default, the system will begin reclaiming buffers when **90** percent of the buffers have been allocated.

Low water mark percentage
: This figure determines at what point, in terms of a percentage of the cache device's total number of buffers, the system stops reclaiming buffers. By default, once the system has reached the high-water mark, above, and begun reclaiming buffers, it will continue to reclaim buffers until fewer than **10** percent are allocated.

Ideally, a cached disk provides a performance improvement by satisfying disk accesses using much faster memory accesses. There are two obstacles, however, that prevent disk caching from reaching this ideal level of performance:

- The cache device is not large enough to contain all the data that applications will require of it; therefore, some I/O requests will require accessing the back end device for data not currently in cache. The ratio of I/O requests satisfied by the front end cache (*cache hits*) to the total number of I/O requests is called the *cache hit rate*. You want the cache hit rate, which is expressed as a percentage, to be as high as possible.

- In a cache used for writing as well as reading, the cache must at some point write, or *flush*, the data in its buffers to disk. If your application accesses the cache during a flush, or if your application causes a flush, it will have to wait, or *stall*, until the flush completes. You want stalls to occur as seldom as possible.

By experimenting with the various parameters, you can find ways to maximize the cache hit rate and minimize the frequency of stalls for your cached disk. Once you have created the cached disk, use Display Cache Statistics to review performance statistics, and use Change Cache Characteristics to adjust the operating parameters.

The operations for managing cached disks are discussed in the following sections.

Creating a Cached Disk

To set up a cached disk on your system, follow these steps:

1. In **sysadm**, execute the operation System –> Kernel –> Build to build a new kernel. When you edit the system file, add these entries:

   nvrd() This entry is for the RAM device at the default location, which you may also specify as **nvrd(0)**. If there are additional **nvrd** devices, they are **nvrd(1)**, **nvrd(2)**, and so on. This entry must appear *after* all disk entries in the system file.

   cdm() This entry is the driver for the cached-disk manager.

   After building and installing the kernel, boot it.

2. After the system has booted, invoke **diskman** and execute the Create Cached Device operation. The operation prompts for the following information:

   `Name` Enter a name for the cached device. This name may be any name that you choose. This will be the name of the entries in **/dev/cdsk** and **/dev/rcdsk**.

   `Front end device`
   Specify the name of a RAM-based raw device such as **nvrd()** or **ldm(***logical_disk***)**, where *logical_disk* is a logical disk built on an **nvrd()** device.

   `Back end device`
   Specify the name of a raw device such as **sd(dgsc(),0)** or **ldm(***logical_disk***)**, where *logical_disk* is a logical disk built on a hard disk. Typically, this is an **ldm()** device containing the name of a logical disk, for example, **ldm(cached_ldu)**.

   `Policy` Select a cache policy. The choices are:

   `Local (all data)`
   You do not intend to export the cached disk for mounting by remote systems, and you want the front end device to be used for user data as well as system data. System data includes file system metadata such as inodes, directory headers, and so on.

`Local (system data only)`
: You do not intend to export the cached disk for mounting by remote systems, and you want the front end device to be used only for system data.

`NFS (all data)`
: You intend to export the cached disk for mounting by remote systems, and you want the front end device to be used for user data as well as system data.

`NFS (system data only)`
: You intend to export the cached disk for mounting by remote systems, and you want the front end device to be used only for system data.

`Pass through`
: Pass data through the front end device directly to the back end device without caching it. This policy effectively turns caching off and allows I/O access to the back end device to occur in the normal manner.

To build a file system on the cached disk, select the **sysadm** operation File System –> Local Filesys –> Create and build the file system on the cached disk's node in the **/dev/cdsk** directory. To access the cached disk as a raw device, use the node in the **/dev/rcdsk** directory.

The Create operation sets the operating parameters for the cached disk to the defaults. To change these parameters, see "Changing Cache Characteristics."

Deleting a Cached Disk

To delete a cached disk and make the front end and back end devices available again as normal raw devices, use the Delete Cached Device operation. You cannot delete a cached device that is open, registered, or mounted.

Deleting a cached disk causes the front end device to flush any dirty buffers to the back end device.

Displaying a Cached Disk

To display information about a cached disk, use the Display Cached Device operation. This display includes the name of the cached device and the cache policy. For a discussion of cache policies, see "Creating a Cached Disk."

The display also includes the values for the various cache operating parameters discussed previously, in "Tuning a Cached Disk."

Displaying Cached Disk Statistics

To display the statistics that the system maintains for your cached disk, select Display Cached Device Statistics. The display shows the following information:

`Total read count`
: The total number of read requests to the cached disk (whether satisfied by buffers in cache or out of cache).

`Total write count`
: The total number of write requests to the cached disk (whether satisfied by buffers in cache or out of cache).

`Percent buffers dirty`
: Percentage of the total number of buffers on the front end device that have been updated but not flushed to the back end device.

`Cache stall rate`
: The average number of times per second that an I/O request has had to wait, or *stall*, while a flush completes.

`Read hit percent`
: The percentage of all read requests that have been satisfied by buffers in cache.

`Write hit percent`
: The percentage of all write requests that have been satisfied by buffers in cache.

`Percent buffers allocated`
: The percentage of the total number of buffers on the front end device that are currently allocated.

See "How Caching Works" and "Tuning a Cached Disk" for more information on how these statistics reflect the efficiency of your cached disk.

Changing a Cached Disk's Characteristics

To modify the values of a cached disk's operating parameters, use the Change Cache Characteristics operation. For a discussion of these parameters, see the earlier section, "Tuning a Cached Disk."

Registering or Deregistering a Cached Disk

Typically, the system registers a cache device at boot and deregisters it when you halt DG/UX with the **halt** command. Registering a cache device makes it available so you can access it like any other raw device. Deregistering a cache device causes it to flush any dirty buffers to the back end device before becoming inaccessible.

If the system fails without performing a normal halt, and if the cache device still contains buffers marked as dirty, the cache device may, if still attached to the same CPU, attempt to flush its buffers to disk when next registered. If the cache device detects that it is attached to a different CPU, it will not flush its buffers; instead, registration fails and the device remains inaccessible until you register it with the Register Cached Device operation or recreate the cache system with Create Cached Device.

Managing Failover Disks

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

If you have a disk-array subsystem, see your hardware documentation for installation instructions. If you simply want to configure two systems to share a common SCSI bus, see the following section.

Sharing a SCSI Bus

This section tells how to set up two computers to share a SCSI bus. This feature is available only on Data General AViiON systems that support second-generation SCSI (such as **ncsc** and **dgsc**) adapters. This form of the dual-initiator configuration has the following benefits:

- You can make more efficient use of seldom-used devices. The two systems can share devices such as tape drives that both use from time to time but which neither requires for constant use.
- You can maintain high availability of critical data. If a hang or a panic occurs on a system currently used to access important information, the second system can take control of the disk by performing a *trespass* and make the data available again with a minimum of downtime.

The basic requirement for a shared SCSI bus configuration is that every device on the bus must have a different SCSI ID. Implied in this requirement is that the SCSI adapters themselves (one on each system), must have different SCSI IDs as well.

CAUTION: *If two or more devices on the SCSI bus have the same SCSI ID, either or both systems may hang or behave erratically. Be sure that each device on the bus, including each SCSI adapter, has a unique SCSI ID.*

To set up two systems to share a SCSI bus, follow these steps:

1. For each device that will be on the shared SCSI bus (other than the SCSI adapters themselves), set the jumpers to indicate a unique SCSI ID. Do not set any devices to SCSI ID 6 or 7; you should leave these numbers unassigned so that you can assign them instead to the SCSI adapters later.

   For example, consider the scenario where you are setting up systems **system1** and **system2** to share a SCSI bus. The systems will share a total of two tape drives and three disk drives. Each device has a unique SCSI ID, as follows:

Device	SCSI ID
1.4GB disk	0
1.4GB disk	1
1.4GB disk	2
QIC–320 tape	4
QIC–320 tape	5

2. Install the hardware: connect the devices to either of the two systems in the standard fashion, such that you can boot and set up both systems independently. The disk that you intend to use for failover may be connected to either system. Do not connect the two systems' SCSI buses together at this time.

In the example configuration, you connect the systems so that **system1** has two disks and one tape drive, and **system2** has one disk drive and one tape drive, as shown in Table 7–2.

Table 7–2 Sample Dual–Initiator Configuration

system1	**system2**
`sd(dgsc(),0)`	`sd(dgsc(),1)`
`sd(dgsc(),2)`	`st(dgsc(),5)`
`st(dgsc(),4)`	

The second disk attached to **system1**, the one at SCSI ID 2, will be the failover disk.

3. Select one of the systems to set up first. Power up and boot the system. If necessary, install the DG/UX system software. This is the system whose SCSI adapter will have the SCSI ID of 6. You set the adapter's SCSI ID when you build the kernel, discussed below.

4. Execute the **sysadm** operation System –> Kernel –> Build to build a new kernel. If you intend to share tape drives between the two systems, make sure device entries for both appear in the system file. Do not, however, add an entry for a failover disk to both systems.

   CAUTION: Be careful to include the failover disk device entry only *in the system file of the system where it is installed. If you include the device entry for a failover disk in the system files for* both *systems, the systems will enter a "race" condition upon booting, and only one of the systems, the one that configures and registers the disk first, will be able to access the disk.*

   Make sure the entries in the system file include the correct SCSI ID for the device as well as the SCSI ID for the local SCSI adapter, SCSI ID 6. Build and install the kernel.

   In the example configuration, boot **system1** and execute System –> Kernel –> Build. You see the following entries in the system file:

   ```
   sd(dgsc(),0)
   sd(dgsc(),2)
   st(dgsc(),4)
   ```

   You want to be able to share the tape drive attached to **system2**, so you add an entry for it to **system1**'s system file:

   ```
   st(dgsc(),5)
   ```

   Then add the SCSI adapter address and SCSI ID in all of these entries. The SCSI adapter address is 0, and the SCSI ID is 6. The list in the system file now looks like this:

   ```
   sd(dgsc(0,6),0)
   sd(dgsc(0,6),2)
   st(dgsc(0,6),4)
   st(dgsc(0,6),5)
   ```

   You proceed with the Build operation and install the new kernel. For more information on building a kernel, see Chapter 4.

5. Shut down the first system. While it is in the SCM, change the SCM's default boot path so that it includes the correct SCSI adapter SCSI ID. For example, if the default boot path had

been **sd(dgsc(),0)**, change it to **sd(dgsc(0,6),0)**. If you also intend to set your system's boot path using **dg_sysctl**(1M) after you have brought the system back up, remember to specify the correct SCSI adapter SCSI ID there as well.

CAUTION: *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.*

6. After resetting the SCM boot path, power off the system.

7. Power on and boot the second system. If necessary, install the DG/UX system software. This system's SCSI adapter will have SCSI ID 7.

8. Execute System -> Kernel -> Build and perform essentially the same steps that you performed for the first system: add device entries for any tape drives connected to the other system that you want to share; then in each device entry set the SCSI ID for the SCSI adapter. On this system, the SCSI adapter SCSI ID is 7.

   For example, consider the sample scenario presented previously. You proceed like this:

   You execute System -> Kernel -> Build. The system file initially contains these entries:

   ```
   sd(dgsc(),1)
   st(dgsc(),5)
   ```

   You want to share the tape drive connected to **system1**, so you add an entry for it in your system file:

   ```
   st(dgsc(),4)
   ```

   Then add the SCSI adapter address and SCSI ID in all of these entries. The SCSI adapter address is 0, and the SCSI ID is 7. The list in the system file now looks like this:

   ```
   sd(dgsc(0,7),1)
   st(dgsc(0,7),5)
   st(dgsc(0,7),4)
   ```

   You now proceed with the Build operation and install the kernel. For more information on building a kernel, see Chapter 4.

9. Shut down the second system. While it is in the SCM, change the SCM's default boot path so that it includes the correct SCSI adapter SCSI ID. For example, if the default boot path had been **sd(dgsc(),1)**, change it to **sd(dgsc(0,7),1)**. If you also intend to set your system's boot path using **dg_sysctl**(1M) after you have brought the system back up, remember to specify the correct SCSI adapter SCSI ID there as well.

   CAUTION: *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.*

10. After resetting the SCM boot path, power off the system.
11. With both machines powered off, reconnect the hardware in the desired configuration: using normal SCSI cables, connect the two systems and the devices as a single SCSI bus. Unlike a typical SCSI bus, this bus has no external terminators on it; the adapters in the systems at each end terminate the bus. Refer to your hardware documentation before performing this step.
12. Power on and reboot the first system with its new kernel.
13. When the first system has completed booting, power on and reboot the second system with its new kernel.

Table 7–3 shows the initial configurations of the example systems.

Table 7–3 Sample Dual-Initiator Configuration

Device	SCSI ID Number	Name Relative to system1	Name Relative to system2
SCSI adapter on **system1**	6	**dgsc(0,6)**	Not accessible
SCSI adapter on **system2**	7	Not accessible	**dgsc(0,7)**
1.4GB disk	0	**sd(dgsc(0,6),0)**	Not accessible
1.4GB disk	1	Not accessible	**sd(dgsc(0,7),1)**
1.4GB disk	2	**sd(dgsc(0,6),2)**	Not accessible
QIC-320 tape drive	4	**st(dgsc(0,6),4)**	**st(dgsc(0,7),4)**
QIC-320 tape drive	5	**st(dgsc(0,6),5)**	**st(dgsc(0,7),5)**

After completing the steps, users on the two systems may share both tape drives, and you may use the disk at SCSI ID 2 as a failover disk. Initially, the failover disk (SCSI ID 2) is accessible only on **system1**. After failing the disk over to **system2**, it is available only on **system2**.

Sharing Tape Drives

Sharing a tape drive is essentially the same in a dual-initiator configuration as on a single system: the tape drive is available for anyone until someone opens it, at which time it remains occupied until the user closes it. The commands that you typically use to open a tape drive include **cpio**, **tar**, and **dump2**. Any attempt to open an already-opened shared tape drive results in the same error that you might see when two users on the same system attempt to access a normal (nonshared) tape drive at the same time. Thus users take turns in sharing a tape drive on the shared bus the same way they take turns when sharing a normal one.

Caveats

If either system boots while the other is using a tape device on the shared SCSI bus, I/O operations to the tape may fail. If a user or application on the running system is using a shared tape device when the other system boots, one of two things will occur: the user or application using the tape drive will receive an I/O error; or the booting system will be unable to configure the tape device. To avoid these problems, make sure no shared tape drives are in use when booting either system.

Using Failover Disks

The section first discusses the principles of operation for failover disks. Then it discusses the tasks involved in managing failover disks.

The DG/UX failover feature allows you to connect a physical disk to two systems and transfer control of the disk from one system to the other as needs arise. The failover feature not only handles operations involved in shutting off access to the disk on one system, but it also handles operations involved in making the disk accessible on the other system and starting any applications used to access it.

The systems that will be using the failover disks must be connected by a TCP/IP network. The failover daemons on the systems communicate over a port running a SAF **listen** port service added during setup of the DG/UX system software.

Only one system may configure and register a physical disk at a time; therefore, a failover disk can be accessible only on one system at a time. When you invoke the operation to transfer control of a failover disk from one system to another, the system performs several tasks.

The system giving the failover disk performs the following tasks if it is functioning normally:

1. Terminate processes accessing the disk.
2. Unmount file systems on the disk.
3. Deregister the disk.
4. Deconfigure the disk.

If the system giving the failover disk is not functioning normally, for example, if it has crashed, you may still perform the failover from the other host by *trespassing*. Trespassing involves forcing the failover to proceed without performing the usual tasks on the system giving the disk.

On the system taking the failover disk, the tasks are:

1. Configure the disk.
2. Register the disk.
3. Call **fsck**(1M) to check file systems.
4. Mount and export file systems.
5. Start applications used to access the disk.
6. Log and report operation status.

The effect of these tasks varies depending on how you have configured the systems. For example, if you have not built file systems on the disk, the steps related to file systems do not occur.

Depending on how you have configured the disks on your system, failover may involve multiple physical disks. This is true if the logical disks or file systems designated for failover span multiple physical disks, as in the following cases:

- A file system built on a multi-piece logical disk whose pieces are on multiple physical disks.
- A striped logical disk, which necessarily resides on multiple physical disks.
- A software-mirrored disk whose images reside on multiple physical disks.

In any of the cases above, the system requires that all such interdependent physical disks be failed over together. The system does not allow you to perform failover such that only part of a logical disk or mirror is being moved to the other system; either all parts of the logical disk or mirror failover, or the operation fails.

Depending on how you have set up your system, there may also be other cases where a failover involves multiple physical disks. For example, you may have a database application in a file system on one physical disk and the database itself located on another physical disk. In this case, you set up both disks for failover and move them together.

The **sysadm** operations that support the failover feature are in the Device –> Disk –> Failover menu. The **sysadm** operations call the **admfailoverapplication**(1M), **admfailoverdisk**(1M), **admfailovergiveaway**(1M), **admfailoverhosts**(1M), and **admfailovertakeaway**(1M) commands. See the manual pages for more information.

To use the failover feature on your systems, perform the following steps for each physical disk that you may like to use as a failover disk:

1. Configure the disk with the **sysadm** operation Device –> Disk –> Physical –> Configure.
2. Register the disk with Device –> Disk –> Physical –> Register.
3. Add the disk as a failover disk with the Device –> Disk –> Failover –> Add operation, being sure to select the Synchronize Database option. A later section discusses the Add operation in more detail.

The failover databases contain a variety of information. Most importantly, they describe the logical disks and file systems on the failover disk, including information required to mount the failover disk's file systems once they become accessible after a failover. The operations that you use to maintain these databases, Add, Modify, and Delete, offer the option of synchronizing databases. If you do not synchronize databases, the system flags them as out of sync, or NOT-SYNC, and will not allow you to transfer control of any failover disk. As an alternative to the Synchronize Database option in the Add, Modify, and Delete operations, you can synchronize databases with the Sync operation. A later section discusses the Sync operation.

NOTE: An attempt to transfer control of a failover disk will fail if databases are not in sync. Synchronize databases with the Sync operation.

With failover disks added and databases synchronized, you are ready to transfer control of a failover database should the need arise. There are several ways to perform failover. When the giving system, the one currently using the disk, is functioning normally, you can perform failover in either of the following ways:

- On the giving system, transfer control of the failover disk to the taking system using the Device –> Disk –> Failover –> Give operation.
- On the taking system, take control of the failover disk using the Device –> Disk –> Failover –> Take operation.

When the giving system is not functioning normally, you transfer control to the taking system by using the Device –> Disk –> Failover –> Take operation with the Trespass option.

Adding Failover Disks

Before you can transfer control of a disk from your system to another, you must add the disk using the operation Device –> Disk –> Failover –> Add. The operation adds an entry for the named local disk and remote host to the local giveaway database. It also adds an entry to the local failover hosts database (not to be confused with the hosts database used by TCP/IP).

The Add operation requires that you supply the following information:

`Local disk module specification`
: Enter the name of a local physical disk, for example, **sd(dgsc(0,6),1)**, that is currently configured and registered. The Device menu contains the Configure operation, and the Device -> Disk -> Physical menu contains the Register operation.

`Remote disk module specification`
: Enter the name that the physical disk will have on the remote system, for example, **sd(dgsc(0,7),1)**.

`Name of host to failover disk to`
: Enter the host name of the remote system. The system must be currently accessible on the network.

`Synchronize database`
: Select this option to copy the information for this failover disk to the takeover database on the remote system. If the remote system is inaccessible, synchronization will fail. When the remote system becomes accessible again, synchronize databases with the Sync operation, discussed later. Until you synchronize databases, the system considers the local giveaway database as out of sync, or NOTSYNC, and restricts you from giving *any* failover disks to other systems.

`Application startup command line`
: Enter the command name to be executed on the remote system after transferring the failover disk. Typically, this command line starts the application used on the remote host to access the data on the failover disk.

The giveaway database includes information normally associated with a local file system, for example, export options, **fsck** pass number, and so on. The Add operation derives this information from the local file system table (**fstab**(4)), which you may review with the File System -> Local Filesys -> List operation. To change this information in the giveaway database, select Device -> Disk -> Failover -> Modify, discussed later.

The Add operation scans the failover disk for dependencies on other physical disks. A failover disk is dependent if it contains incomplete logical disks or mirrors with parts residing on another physical disk. If any such dependencies exist, the Add operation displays a warning. You cannot transfer control of a failover disk without also transferring any codependent disks with it.

Deleting Failover Disks

To remove a failover disk from the local giveaway database, select the operation Device -> Disk -> Failover -> Delete. If this physical disk has dependencies on other physical disks, the operation will warn you. A dependent physical disk is one that contains part of a logical disk or mirror that also has parts residing on other physical disks. You cannot failover a disk without also failing over any disks that have dependencies on it.

The Delete operation requires the following information:

`Host name`
: Select the name of the remote host designated for the failover disk.

`Disk name`
: Select the name of the local physical disk designated as the failover disk.

`Synchronize database`
: Select this option to remove the information for this failover disk from the takeover database on the remote system. If the remote system is inaccessible, synchronization will fail. When the remote system becomes accessible again, synchronize databases with the Sync operation, discussed later. Until you synchronize databases, the system considers the local giveaway database as out of sync, or NOTSYNC, and restricts you from giving *any* failover disks to other systems.

Modifying Failover Disks

To change the information about a failover disk in your giveaway database, select the operation Device –> Disk –> Failover –> Modify. You may, for example, wish to change some of the file system mounting information (such as mount point, export options, and so on) associated with a file system on the failover disk.

The Modify operation requires the following information:

`Host name`
: Select the remote host designated for the failover disk.

`Disk name`
: Select the physical disk used for failover. You may select **all** disks.

`LDU name`
: Select the pathname of the logical disk's special file. For example, the special file for a logical disk named **db1** would be **/dev/dsk/db1.**

The operation then allows you to change any of the other information associated with the logical disk entry, including the remote disk name and the file system mounting information. For complete discussion of file system mounting information, see the discussion of local file systems in Chapter 8.

The operation also offers the `Synchronize database` option, which you select to enter this modification in the takeover database on the remote system. If the remote system is inaccessible, synchronization will fail. When the remote system becomes accessible again, synchronize databases with the Sync operation, discussed later. Until you synchronize databases, the system considers the local giveaway database as out of sync, or NOTSYNC, and restricts you from giving *any* failover disks to other systems.

Listing Failover Disks

To display information about local failover disks, select the Device –> Disk –> Failover –> List operation. The display includes information for failover disks on the local system, stored in the giveaway database, as well as remote hosts used for failover, stored in the hosts database (not to be confused with the network hosts database used by TCP/IP).

The information for the hosts database includes the remote system name, the path used for communication with the remote system's failover daemon, and the synchronization status of the remote takeaway database. If the remote takeaway database is out of sync, or NOTSYNC, the remote system cannot give *any* failover disks to another system. You synchronize databases with the Sync operation, discussed later.

The information for the giveaway database includes the following information for each logical disk piece on each failover disk:

`Hostname`
: The remote host designated for failover.

`Local Diskname`
: The local name of the failover disk on which the piece resides.

`Remote Diskname`
: The remote name of the failover disk on which the piece resides.

`File System Source`
: The pathname of the special file for the logical disk, for example, **/dev/dsk/db1**.

`Pce Nbr`
: Of the total number of pieces making up this logical disk, this is the piece number for this piece.

`Tot Nbr`
: This is the total number of pieces making up this logical disk.

`<FS INFO>`
: The information used for mounting the file system, such as mount point directory, **fsck** pass number, and so on. For a discussion of this information, see the discussion of local file systems in Chapter 8.

Giving away a Failover Disk

To transfer control of a failover disk from your system to another, select the operation Device –> Disk –> Failover –> Give. The operation requires that you enter the host name of the remote host and the names of the failover disks you wish to give. You should have already added the failover disks with the Add operation.

The process of giving a failover disk involves having the local failover daemon, **failoverd**(1M), consult the giveaway database and perform a variety of actions. The discussion of failover concepts, earlier in the chapter, discusses these actions.

You cannot give *any* failover disk if your local giveaway database is out of sync, or NOTSYNC. Synchronize databases with the Sync operation, discussed later.

Taking away a Failover Disk

To transfer control of a failover disk from another system to your own, select the operation Device –> Disk –> Failover –> Take. The operation requires that you enter the host name of the remote host and the names of the failover disks you wish to take. You can take a failover disk only if

- the administrator of the remote system has added the disk using the Add operation, and
- the failover databases on your system have been successfully synchronized with the databases on the remote system.

The process of taking a failover disk involves having the local failover daemon, **failoverd**(1M), consult the takeaway database and perform a variety of actions. The discussion of failover concepts, earlier in the chapter, discusses these actions.

The operation requires that you specify the host name of the remote system and the local name of the failover disk.

Optionally, you may select the Use Trespass option. This option forces the Take operation to proceed even if the remote system does not respond. This option is intended for situations where the remote system is hung or crashed. If the remote system is functioning normally, do not select the Use Trespass option. If you select the Use Trespass option when taking a failover disk from a normally functioning system, access to the failover disk on the remote system will be terminated ungracefully, possibly resulting in unnecessary data loss.

You cannot take a failover disk if your local takeaway database is out of sync, or NOTSYNC. Synchronize databases with the Sync operation, discussed later.

Verifying the Failover Disk Database

To make sure that the local giveaway database has up-to-date information about failover disks and their logical disks, or to list physical disk dependencies, select the operation Device -> Disk -> Failover -> Check.

The operation requires that you specify the host name of the remote system, the name of the failover disk, and the database files whose entries you would like to verify. The two files are the giveaway database, which contains information about physical and logical disks designated to be given to other systems, and the takeaway database, which contains entries for physical and logical disks designated to be taken from the other systems.

The operation scans the physical disk for all logical disk pieces and correlates them with their entries in the local file system table, **fstab**, which you can display with the File System -> Local Filesys -> List operation. The operation then updates entries as necessary in the selected failover databases.

The operation also scans the physical disks for dependencies on other physical disks. A physical disk is considered dependent when it contains parts of logical disks or mirrors that have parts on other physical disks. The operation reports any such dependencies that it finds. You may then add these other physical disks, using the Add operation, as necessary.

Synchronizing Failover Databases

To make the failover databases on your system consistent with those on a remote system, select the operation Device -> Disk -> Failover -> Sync. For the specified remote system, the operation scans the local giveaway database and copies to the remote takeaway database entries for any logical disks designated to be given to the remote system. Conversely, the operation scans the takeaway database on the specified remote system and copies to the local giveaway database entries for any logical disks designated to be taken by this system.

The operations that change the local giveaway database, Add, Modify, and Delete, offer you the option of synchronizing databases. This option causes the operation to copy the changes as appropriate to the takeaway database on the remote system. If you do not synchronize databases, the system flags the local giveaway database as out of sync, or NOTSYNC, and restricts you from giving failover disks to another system until you have synchronized databases.

Managing a File System

Managing file systems includes tasks such as creating and deleting them, changing their size, and checking their internal consistency. In **diskman**, these operations are in the File System Management Menu. In **sysadm**, these operations are under **sysadm**'s File System menus, which also contain a number of other file system-related operations. For information on these other file system-related operations, see Chapter 8.

Making a File System

To create a new, empty DG/UX file system on a logical disk select the **sysadm** operation File System –> Local Filesys –> Create. In **diskman**, select Make a File System from the File System Management Menu.

The **sysadm** operation calls the **admfilesystem**(1M) command to perform the necessary tasks. Both **admfilesystem** and the **diskman** operation ultimately call the **mkfs**(1M) command to create the file system.

The operation first prompts you for a logical disk name. After you have specified a logical disk for the file system, the operation prompts you for any **mkfs** flags you may want to add. See the **mkfs** manual page for complete information. Briefly, **mkfs** provides options for setting the various attributes of a file system. In most cases, the default values are sufficient. If you intend to use the file system to store unusually large files or an unusually large number of files, however, you may find that you can help I/O performance by adjusting some of the default file system attributes. See the **mkfs**(1M), **fs**(4), and **tunefs**(1M) manual pages for detailed information on file system internal structure. Your Data General representative may have additional information on DG/UX file system structure.

To create an MS-DOS file system, see Chapter 15.

When planning a logical disk's size, you need to consider file system overhead for the kind of file system (if any) you intend to put on the logical disk. File system overhead refers to internal data structures, such as data allocation tables and so on, that the operating system requires in order to manage file access in the file system.

When planning a logical disk for a DG/UX file system, you need to make it at least 17% larger than the amount of space you intend to use in the file system. This 17% overhead includes the 10% reserved free space buffer and the internal structures that the operating system requires for tracking files and directories. For example, if you need a file system large enough to hold 100 MB of data, you should create a file system 117 MB in size. With no files, the file system will be around 4% full. After adding 100 MB of data, the file system will be around 90% full. When a file system becomes 90% full, only the superuser can create or extend a file or directory in the file system.

After creating the file system, you need to add an entry to the file system table with the **sysadm** operation File System –> Local Filesys –> Add.

Next, make the file system available on a directory so users can access it. Use the **sysadm** operation File System –> Local Filesys –> Mount.

Licensed Material – Property of copyright holder(s)

Checking a File System

To check a file system for inconsistencies, select the **sysadm** operation File System –> Local Filesys –> Check or the **diskman** operation Check a File System, located in the File System Management Menu. The **sysadm** operation invokes the **admfilesystem**(1M) command, and both **admfilesystem** and **diskman** invoke **fsck**(1M) to perform the check.

The **fsck** program checks blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region information. The **fsck** program reports any inconsistencies within the file system. Depending on the options you specified for **fsck**, it may also fix these inconsistencies. When **fsck** runs automatically at boot time, a copy of its output goes to **/etc/log/fsck.log**. When **fsck** finds a fragment of a file and cannot restore it to its original file, it places the fragment in a **lost+found** directory created for the purpose in the file system's mount point directory. Chapter 2 suggests ways to investigate file fragments and ascertain what they are and to whom they belong.

You can invoke **fsck** to check any file system, whether mounted or unmounted, but it can only repair a file system if it is unmounted.

For a detailed discussion of **fsck**, see Chapter 8 and the **fsck**(1M) manual page.

Expanding a File System

To increase the size of an existing file system and its logical disk, select the **sysadm** operation File System –> Local Filesys –> Expand or the **diskman** Grow operation from the File System Management Menu.

Use File System –> Local Filesys –> Unmount to unmount the file system before performing this operation. Remote systems that have mounted the file system should also unmount it. Whether they unmount it or not, they will have to unmount it and remount it eventually to restore access to the file system after the operation.

The operation prompts you for the name of the logical disk, the starting address for the new space, and the size of the new space in 512-byte blocks. If the new space follows an existing piece of the logical disk, the operation simply extends the existing piece to provide the new space. If the new space does not follow an existing piece, the operation allocates the space as a new piece of the logical disk.

If expanding the root (**/**) or **/usr** file systems, do not permit the operation to continue if it will add a piece to the logical disk. Before you grow the root (**/**) or **/usr** file systems, remember that you cannot boot from a file system built on a multi-part logical disk. 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 diskman, **/usr/stand/diskman**, which you may need to boot to recover after a failure. To increase the size of either of these file systems, first rearrange the contents of the physical disk so that the desired free space is located immediately after the file system, thus allowing the operation to expand the file system by stretching the current piece rather than creating a second piece.

You may continue to add pieces to the logical disk until it has 32 pieces.

Keep in mind that file system internal data structures and the free space requirement (10% by default) will use some of the space that you are adding to the file system. Consequently, you should add 17% more space than the amount you intend to use. For example, if you need 30,000 more blocks of usable space in a file system, you should add 30,000 + (17% of 30,000) blocks, or 35,100 blocks.

Striped and Software-Mirrored Logical Disks
To change the size of a striped logical disk, you must create a new, larger striped logical disk and then copy the original striped logical disk's contents to it. You may have to copy the original striped logical disk's contents to tape first to make room for the new, larger striped logical disk.

To change the size of a mirror, first unmount the mirror. Then use Device -> Disk -> Software Mirror -> Break to break it apart so that its component images are available as individual logical disks. Expand the logical disks separately, then recreate the mirror. When you recreate the mirror, you will have to synchronize all but one of the images (the one you use as the master for synchronization).

Shrinking a File System

To decrease the size of an existing file system and its logical disk, select the **sysadm** operation File System -> Local Filesys -> Shrink or the **diskman** operation Shrink a File System, located in the File System Management Menu.

You must unmount the file system before performing this operation. Use File System -> Local Filesys -> Unmount. Remote systems that have mounted the file system should also unmount it. Whether they unmount it or not, they will have to unmount it and remount it eventually to restore access to the file system after the operation.

The operation prompts you for the number of 512-byte blocks that you want removed from the end of the file system. The operation selects which blocks to remove based on internal criteria.

The Shrink operation maintains the required percentage of free space for the file system. By default, a file system has a free space requirement of 10%.

Shrinking a file system requires the operation to rearrange data within the file system, collecting unallocated blocks so that the system may free the desired number. The internal structures of a file system complicate this process even more, particularly because some of these internal structures, like the file system itself, are designed to contain a certain amount of free space. The shrink operation will not transgress this built-in cushion of free space because doing so may result in poor I/O performance for the file system. If you specify that the shrink operation recover more space than is available from the file system, it returns an error.

To give you some flexibility, the shrink operation allows you to specify two file system parameters for the shrink operation:

`Percentage of blocks in the remaining DARs to keep unallocated`
: You may set what minimum percentage of the file system should remain empty after the shrinking operation. This figure refers to the percentage of unallocated blocks per DAR (Disk Allocation Region). The default is 20%. If the shrink operation cannot free the desired number of blocks while leaving this percentage of remaining space free, the operation fails.

`Percentage of file node slots in the remaining DARs to keep available`
: You may set what minimum percentage of file node slots should remain available per DAR after the shrinking operation. A DAR can hold a fixed number of files and direc-

tories (file nodes), and this percentage determines how many of them will remain available for directory or file creation. The default is 20%. If the shrink operation cannot free the desired number of blocks while leaving the desired percentage of file node slots available, the operation fails.

Under some conditions, the Shrink operation will reduce the file system more than the requested number of blocks. This happens when the Shrink operation causes the new end of the file system to fall in a DAR's metadata area (file node table or free space bitmap). The operation detects this condition and shrinks the file system further, to the beginning of the DAR. This condition does not constitute an error or a problem. When it occurs, **diskman** prints the following message:

```
The last DAR size was less than the minimal needed to support
the file node table and the DAR bitmap.  The new size of the
file system has been adjusted to nnnn blocks.
```

For a more detailed discussion of the file system internals, see the **mkfs**(1M), **fs**(4), and **tunefs**(1M) manual pages.

Striped and Software-Mirrored Logical Disks
To change the size of a striped logical disk, you must create a new, smaller striped logical disk and then copy the original striped logical disk's contents to it. You may have to copy the original striped logical disk's contents to tape first to make room for the new, smaller striped logical disk.

To change the size of a mirror, first unmount the mirror. Then use Device –> Disk –> Software Mirror –> Break to break it apart so that its component images are available as individual logical disks. Shrink the logical disks separately, then recreate the mirror. When you recreate the mirror, you will have to synchronize all but one of the images (the one you use as the master for synchronization).

Managing Disk–Arrays

The various disk-array subsystems have different utilities for managing them. If you have a HADA–I subsystem, for example, use the Device –> Disk –> Manage Array operation, which invokes **/usr/sbin/gridman**. See your disk-array subsystem documentation for information on invoking the appropriate management utility.

End of Chapter

Chapter 8 File System Management

File system management tasks apply to all systems, even to those that do not have their own disks. File system management involves creating file systems and making them available on the system. It also involves verifying file system consistency, backing up and restoring files, and maintaining the file system table, **/etc/fstab**. The file system table contains entries for the local and remote file systems that you want available on your system.

For a summary of DG/UX system disk- and file system features that can improve disk file service on your system, see Chapter 7, Disk Management.

This chapter gives brief explanations of logical disks, mounting file systems, backup cycles, and shows you how to manage your DG/UX file systems.

File System Terms

You may also want to review the information in Chapter 7, Disk Management, which discusses logical disks, physical disks, and features for improving disk and file system service.

We use the following terms in this chapter:

/etc/fstab The **fstab**(4) file is the file system table describing local and remote file systems that you want accessible on the local system. File systems listed in **fstab** become accessible automatically at boot time. The **fstab** file contains information for commands that mount, unmount, backup, restore, and check file systems.

mount point directory
After you create a logical disk, you usually create a file system on it. Next you mount the file system on a directory. From then on, the name of the file system is the name of the directory where you mounted it, called the mount point directory. When operations such as Backup, Restore, Check, or Disk Use require the name of a file system, you provide the name of the mount point directory.

mount To attach a file system to a directory, making it accessible to users. The system mounts directories listed in **fstab** at boot time. You can mount directories explicitly with the shell command **mount**(1M) or with **sysadm**'s Mount operation. You mount local file systems as well as remote ones. To mount remote file systems, you need the ONC/NFS network software.

unmount To detach a file system from a directory, making it inaccessible to users. The system unmounts remote (ONC/NFS-mounted) file systems when you shut your system down to run level 2 or lower. The system unmounts local file systems (except **/usr** and **/**) when you shut the system down to run level S (single–user mode).

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.

pass number
A number from a file system's **fstab** entry that indicates the order in which the **fsck** program checks file systems for corrupted and damaged files. In the first pass, **fsck** simultaneously checks all file systems with a pass number of 1. In the second pass, **fsck** checks file systems with a pass number of 2, and so on.

read-write mode
Access permission that allows users to write (change) files and directories in the file system as well as read them. This mode does not override the normal permissions that already apply to the files and directories.

read-only mode
Access permission that allows only reading of a file system.

ONC/NFS Network File System. A network software package that allows you to access remote file systems as though the remote file systems resided on a local disk. For more information, see *Managing ONC™/NFS® and its Facilities on the DG/UX™ System.*

export To make a local file system available for mounting by other systems in your network. To export a local file system or mount an exported file system from a remote system, both systems must be running the ONC/NFS software.

fast recovery file system
A file system mounted with the **fsck** logging feature. The system records file system modifications to a log, which **fsck** uses during recovery after a failure. Using the log, **fsck** can check and repair the file system faster than it can without the log.

File System Perspectives

As discussed in Chapter 7, the DG/UX system uses logical disks, a feature that provides more flexibility for organizing file systems. The operating system treats a logical disk as if it were a real physical disk.

You can view file systems in two ways. From the operating system's perspective, a file system is associated with a logical disk, which in turn is associated with sections of physical disks accessed through a device node. From the user's perspective, a file system is a hierarchical directory structure. Logical disk names assigned by a user are incorporated into the nodenames used by the operating system.

The kernel creates device nodes at boot time. These device nodes provide the operating system with access to logical disks. For each device and for each logical disk, the kernel creates a device node each time you boot the system.

The Operating System's View of File Systems

From the operating system's point of view, a file system is associated with sections of one or more physical disks. Logical disks form the bridge between file systems and physical disks. The file system associated with the logical disk is mounted (made available to users) in the directory structure.

You can think of the relationship between file systems, logical disks, and physical disks as a three-level hierarchy:

File system	The level at which the user interacts with the system.
Logical disk	The intermediate level that associates the file system with the physical disk. The file system and the rest of the operating system interact at this level.
Physical disk	The level at which the operating system interacts with the hardware.

Logical disks have other functions besides acting as bridges between physical disks and file systems. The **swap** logical disk, for instance, does not have a file system on it. The **swap** logical disk is an area of a disk that is accessed directly only by the operating system.

The User's View of File Systems

From a user's viewpoint, there appears to be one file system: the single hierarchy consisting of all files and directories on the system. Because all file systems are mounted under the **/** (root) file system, a system's entire directory tree appears to be a single hierarchical directory structure. If you mount a new file system, the directory tree (as seen by the user) expands, starting at the point where the new file system is mounted. The user sees a group of new files under a new directory name. If you unmount a file system, part of the directory tree disappears.

The operating system prevents you from inadvertently unmounting a file system that someone is using. For example, if a user changed directory to **/usr/opt/X11/catman**, you would not be able to unmount the **/usr/opt/X11** file system. Similarly, if a user were executing a program in **/usr/opt/X11/bin** (regardless of the user's working directory), you would not be able to unmount **/usr/opt/X11** until the executing process terminated.

Operations for Creating a Local File System

You can create a file system on a local disk in any of three ways:

- In **diskman**'s File System Management Menu, select the Make a File System operation.
- In **sysadm**, select File System –> Local Filesys –> Create.
- Execute the **mkfs**(1M) command at the shell prompt.

The **diskman** and **sysadm** operations for creating file systems simply call the **mkfs** command. All of the three methods let you specify additional **mkfs** options to tailor the various file system components to fit your needs. A later section discusses some **mkfs** options in more detail.

The "File System Management Procedures" section describes **sysadm**'s Create operation for creating file systems.

Making a Local File System Accessible

Once you have created a file system (File System –> Local Filesys –> Create) and added it to the file system table (File System –> Local Filesys –> Add), you make it accessible to local users by attaching or *mounting* it with File System –> Local Filesys –> Mount.

You mount the file system on a directory of your choice called a *mount point directory*, or just a *mount directory*. The mount directory thus becomes the name you use to refer to the file system. It is good practice to give the mount directory the same name as the logical disk where the file system resides. For example, for a logical disk named **thor**, we have mount directory (and file system name) **/usr/thor**.

When the system changes to certain run levels, it mounts file systems that have entries in **fstab**. You may mount a file system yourself at any time with **sysadm**'s Mount operation. When you mount a file system, the previous contents of the mount point directory become hidden and inaccessible.

Every time you boot your system, the kernel creates new device nodes; therefore, in our example above, the nodename **/dev/dsk/thor** will be created. Figure 8–1 shows mount directory **thor** branching from **/usr**. We now have a new file system named **/usr/thor**.

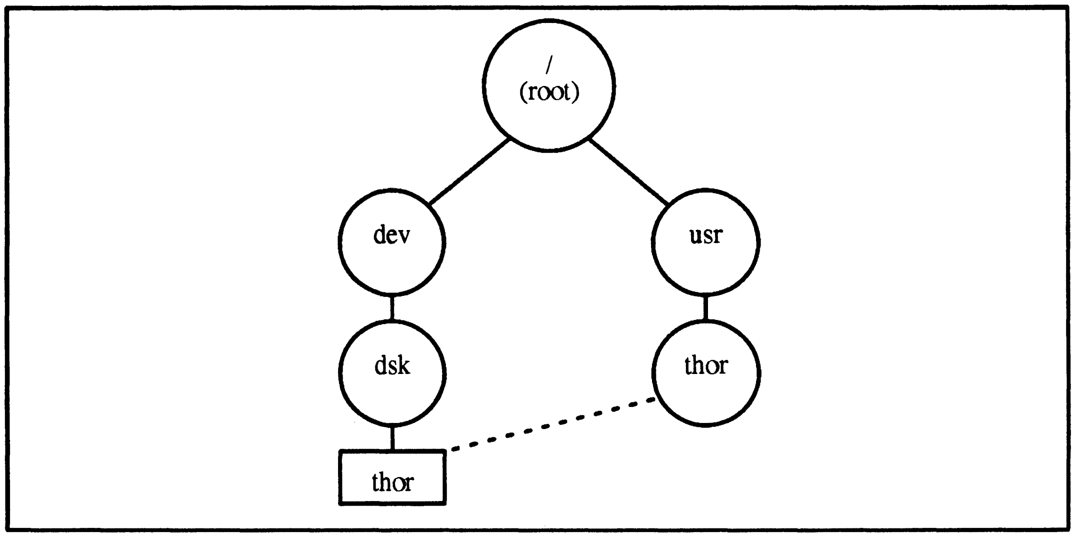

Figure 8–1 Mounting a File System

In Figure 8–1, circles represent directories, and the square represents a file, in this case the special file **/dev/dsk/thor**, which is the logical disk **thor**. The directory **/usr/thor** is the mount point directory where you have mounted **thor**.

You can think of each file system as an independent directory tree. The Mount operation "grafts" a file system tree onto a larger tree. The top of the total tree is the **/** (root) directory, which is also the top directory of the root file system.

As a part of routine management, you or someone else should make backup tapes of your file systems. The DG/UX system offers three predefined backup cycles, which are discussed in "Managing the Backup Cycle." Read this section and review the three backup cycles to see which is most appropriate for your system. You may even design your own backup cycle. To perform the backup, use the operation File System -> Backup -> Create.

File System Management Procedures

Both **sysadm** and **diskman** provide operations for managing file systems. In **sysadm**, the operations are in the File System menu. In **diskman**, the operations (for managing local file systems only) are in the File System Management Menu. The following sections elaborate on the tasks you can perform with these operations.

Backing Up and Restoring 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 quite tedious if you do not remember the exact day when the file was last changed.

The **sysadm** Backup menu provides operations for making backup copies of file systems and for restoring files and file systems from backups. There are also submenus for setting the backup medium (such as the kind of tape or tape device) and for manipulating the backup cycle.

Backing Up Files

You back up systems using the **sysadm** File System –> Backup –> Create operation. 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 Dump Cycle."

Backing Up with sysadm

To make a backup of a file system, select 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 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 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 with dump2(1M)

The **dump2** program copies some or all files on a logical disk to the backup medium based on the backup "level." There are 10 levels: 0 through 9. Execute the **dump2** program by specifying a logical 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 has been "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 Oct 30 23:58:58 1992
```

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 October 30, 1992. 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.

Restoring Files

You can restore files using either the **sysadm** File System –> Local Filesys –> Restore operation 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 (noninteractive) 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 **restore**'s interactive mode are in "Example: Using restore in Interactive Mode," which follows the next section.

Restoring Files 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

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)

The following steps 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 logical 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.

Example: Using restore in Interactive Mode

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** keyletter. 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 '.'? [yn] n↵
```

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 Backup Media

The Medium menu 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. Figure 8–2 shows an example listing.

```
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)
```

Figure 8–2 Example List Output from Medium Menu

The default medium, QIC-150 cartridge, is what appears as the default medium in the Backup menu's Create operation. You may reset this default to any other entry with the Default operation.

Managing the Backup Cycle

Use the 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**
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 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).

Retrieving Information about Files and File Systems

This section shows you how to find and display information about files in your file systems. The **sysadm** File System –> File Information menu has three operations for getting information about files and file systems:

`Disk Use`	This operation displays information about disk space taken up by file systems.
`Check`	This operation searches for files that may constitute a security risk: device files not located in **/dev** and executables owned by the superuser that have the setuid bit set.
`Find`	This operation locates files and directories based on criteria that you specify.

We use the following terms in this section:

inode	Data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID. The number of inodes represents the total number of files that can exist on the system. The **mkfs**(1M) program, which creates a file system, accepts options that you can use to control the number of inodes in a file system. An inode is 126 bytes long, and there are 4 inodes to a disk block.
disk block	A 512-byte unit of data as it is actually stored and manipulated.
setuid	A mode bit that can be specified for any executable file. When a user runs an executable file that has the setuid bit set, the system gives the user the permissions of the owner of the executable file for the duration of the command. See **chmod**(1).
/dev	Administrative directory containing entries for all devices on the system.

Displaying Disk Space Usage

Select the Disk Use operation to display a table showing the number of blocks and inodes in use on mounted file systems that you specify. You may enter multiple file system names. If you specify no file system names, the operation lists information for all file systems. Here is an example of a display that Disk Use provides:

```
             Free     Total    Pct     Free     Total    Pct
Directory   Inodes    Inodes   Used   Blocks    Blocks   Used
---------  -------  --------  ----  --------  --------  ----
/tmp/root     5147      5760   10%     24618     40000   38%
```

Checking for Security Breaches

Select the Check operation 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, as system administrator, have created or moved device files for a special purpose, such as a test.
- Nonsystem 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**(1) 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.

Finding Files

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

The Find operation is useful as a part of everyday maintenance. You can use it to find

- files whose names match a specific pattern,
- files belonging to particular users or groups,
- files of a given type,
- files that have not been accessed or modified in a long time and are no longer necessary, or
- files that take up too much space.

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

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**(1). 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.

Making Tapes

This section does not discuss a **sysadm** menu procedure; it simply offers some suggestions for making tapes. 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 all subdirectories and files under it), do the following:

1. Mount a tape and put the drive on-line.
2. Go to the directory you wish to backup:

   ```
   # cd /sales/smith ↵
   ```

3. Backup everything in the directory to tape:

   ```
   # find . -print | cpio -ocvB > /dev/rmt/0 ↵
   ```

   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.

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.

Managing Local File Systems

Use the Local Filesys menu to create file systems on local disks, manage the file systems, change their size, make them accessible on the local system and on remote systems, and check their internal consistency.

Creating a File System

You can create a file system using **diskman**, **sysadm**, or the **mkfs**(1M) command. This section tells how to use the **sysadm** operation File System -> Local Filesys -> Create.

By default, the system creates DG/UX file systems, but you may also create an MS-DOS file system. To use MS-DOS file systems, the MS-DOS file system manager driver, **dfm**(), must be configured in your kernel.

For creating file systems on a disk, there are two paths from which you can choose:

- You can create system areas and multiple file systems on the disk. This is typically what you do with normal hard disks and magneto-optical disks.

  System areas not only contain data necessary to manage multiple file systems on the disk, but they also provide bad block remapping capabilities. Bad block remapping is the system's automatic way of tracking and avoiding bad blocks on the disk medium. Even if you intend to create only one file system on a disk, this alternative is preferable because of the bad block remapping capability.

- You can create one file system over the entire physical disk without creating system areas. This is typically what you do with diskettes and other small media. You trade the benefits of multiple file system and bad block remapping capabilities for the disk space saved by not creating system areas.

To create system areas on a disk, see **diskman**'s Physical Disk Management Menu. You then use **diskman** to allocate the disk space in the form of logical disks. See Chapter 7 for more information on **diskman**.

When planning a logical disk, you need to consider file system overhead for the kind of file system (if any) you intend to put on the logical disk. File system overhead refers to internal data structures, such as data allocation tables and so on, that the operating system requires in order to manage file access in the file system.

When planning a logical disk that will contain a DG/UX file system, you need to make it at least 17% larger than the amount of space you intend to use in the file system. This 17% overhead includes the 10% reserved free space buffer and the internal structures that the operating system requires for tracking files and directories. For example, if you need a file system large enough to hold 100 MB of data, you should create a file system 117 MB in size. With no files, the file system will be around 4% full. After adding 100 MB of data, the file system will be around 90% full. When a file system becomes 90% full, only the superuser can create or extend a file or directory in the file system.

Once you have created the logical disks, you are ready to create the file systems. To create a file system on a logical disk, select File System -> Local Filesys -> Create.

The operation prompts you to select a logical disk for the file system.

The Create operation allows you to specify options to the **mkfs** command. These options allow you to tailor the various internal data structures of the file system to optimize for performance or other reasons. In most cases, the default file system values are sufficient. If you intend to use the file system to store unusually large files or an unusually large number of files, however, you may find that you can help I/O performance by adjusting some of the default file system attributes. See the **mkfs**(1M), **fs**(4), and **tunefs**(1M) manual pages for detailed information on file system internal structure. Your Data General representative may also have additional information on DG/UX file system structure.

To create an MS-DOS file system, see Chapter 15.

See the **mkfs**(1M) manual page for more information on options.

CAUTION: *Creating a file system destroys all data on the logical disk. If a file system already exists on a logical disk, creating a new file system will destroy the existing one.*

Once you have created a file system, you add it to the system's table of known file systems using the operation File System -> Local Filesys -> Add. Then you make the file system accessible to users with the operation File System -> Local Filesys -> Mount.

Adding a Local File System

The system recognizes only the file systems that have entries in **fstab**. When the system boots, it mounts any file systems that have entries in **fstab**. Select the Add operation to put an entry for a file system into the file system table, **/etc/fstab**.

The Add operation's first query is for the file system type. The accepted types are:

`dg/ux`	This is the typical disk file system.
`ramdisk`	This is a file system that resides entirely in memory, hence the name `ramdisk`, where `ram` stands for random access memory. You do not need to allocate disk space for a `ramdisk` file system. A `ramdisk` file system provides very fast I/O performance for file systems that can fit in memory. The data disappears from memory when you delete or unmount the file system or when the system goes down. See **mfs**(4) for more information.
`cdrom`	This is an ISO 9660 or High Sierra CD-ROM file system. To use an ISO 9660 or High Sierra drive on your system, your kernel must be configured with the

hfm() file system manager driver. For more information, see Chapter 15 and the **hfm**(4) manual page. If you want the system to mount all `cdrom` type file systems when it comes up to run level 1 or higher, edit **/etc/dgux.params** and add **–t cdrom** to the **localfs_ARG** parameter. See Chapter 15 for other CD-ROM considerations.

`dos` This is an MS-DOS file system, created by **mkfs** with the **dos** flag. To use MS-DOS file systems, the MS-DOS file system manager driver, **dfm**(), must be configured in your kernel. See **dfm**(4) for more information. If you want the system to mount all **dos** type file systems when it comes up to run level 1 or higher, edit **/etc/dgux.params** and add **–t dos** to the **localfs_ARG** parameter. See Chapter 15 for other MS-DOS file system considerations.

The file system type that you select determines what queries the Add operation presents. The following sections discuss the various queries. The discussion of each query notes the types of file systems to which it applies.

`Logical Disk`

This prompt appears when you add a `dg/ux` type file system.

This is the name of the logical disk where the file system is located. The logical disk must already exist. If an existing logical disk does not appear in the list, it may be because you did not register the physical device.

`Device File`

This prompt appears when you add a `ramdisk`, `dos`, or `cdrom` type file system.

For `ramdisk` type file systems, this is any arbitrary name that you choose to call the device. The name may be any legal file name. When created, the name appears in the **/dev** directory.

For `dos` and `cdrom` type file systems, this is the name of the physical device node file in **/dev/pdsk**.

`Mount Directory`

This prompt appears when adding all file system types.

This is the directory on your system where you want the file system to appear. If the directory does not already exist, the Add operation will ask if it should create the directory.

`Use Wired Memory`

This prompt appears when you add a `ramdisk` type file system.

By default, the system uses unwired memory for the file system. Unwired memory may be swapped out to the swap area on disk in order to free up needed physical memory. If you select wired memory, the file system stays completely in physical memory without being swapped out. Do not select the wired option unless your system has enough physical memory to hold the entire file system. See **mfs**(4) for more information.

`Maximum File Space`

This prompt appears when you add a `ramdisk` type file system.

Set the maximum number of 512-byte blocks that the file system may use. The system does not attempt to allocate memory until the file system requires it. The maximum amount of memory that the system will allocate for the file system is this maximum or the amount of memory available on the system, whichever is less. Any operation that causes the system to attempt to allocate more than this maximum will fail.

`Maximum File Count`

This prompt appears when you add a `ramdisk` type file system.

Set the maximum number of files that the file system may contain.

`Write Permission`

This prompt appears when you add a `dg/ux` or `dos` type file system.

If the write permission is `Read/Write`, users may change the file system as well as read and execute files in it. This is the normal mode for a file system; it does not allow users to override the normal file system security architecture. If the write permission is `Read Only`, users may only read and execute files in the file system.

`File Permission`

This prompt appears when you add a `dos` type file system.

Specify the permissions mask that will determine the permissions on the files in the file system. A permissions mask is an octal expression of the mode bits that determine how users can access a file.

For a more complete discussion of file modes, see the manual page for **chmod**(1).

`Dump Frequency`

This prompt appears when you add a `dg/ux` type file system.

The dump frequency determines how often the file system backup utility, **dump2**(1M), will back up the file system. The backup utility runs whenever you execute **dump2** explicitly or select the **sysadm** operation File System –> Local Filesys –> Backup –> Create.

There are four kinds of dump frequency:

`Daily`	Back up this file system during every daily, weekly, and monthly backup.
`Weekly`	Back up this file system during every weekly and monthly backup.
`Monthly`	Back up this file system only during monthly backups.
`None`	Do not back up this file system.

You select a dump frequency for a file system based on the importance of the data in the file system and how often you alter it. The schedule that determines when daily, weekly, and monthly backups occur is the backup cycle, described in "Backing Up and Restoring File Systems" earlier in the chapter.

`Fsck Pass Number`

This prompt appears when you add a `dg/ux` type file system.

The **fsck** pass number, a digit `0` – `9`, indicates the pass on which the file system checker, **fsck**(1M) (when invoked with **fsck -p**), should check this file system for damaged or corrupt files. File systems with pass numbers between (and including) `1` and 9 will be checked in order. That is, all file systems with number 1 are checked first, then those with number 2, and so on. The digit 0 indicates that a file system should never be checked. For more information on **fsck**, see "File System Checking: fsck," at the end of this chapter.

`Fsck Logging`

This prompt appears when you add a `dg/ux` type file system.

Select **fsck** logging if you want the file system mounted for fast recovery. With **fsck** logging turned on for a file system, the system logs modifications to the file system. If a failure leaves the file system in a corrupt state, **fsck** can use the log to speed recovery.

Logging is good for file systems where it is crucially important to minimize the amount of time during which the file system is unavailable (as during verification and repair). Because logging has some negative impact on run time write performance in the file system, we recommend it primarily for file systems where rapid recovery and high availability are crucial.

If you select **fsck** logging, the operation later prompts you for log size. Specify the log size in 512-byte blocks. There is a tradeoff in performance between log files of different sizes. A large log file improves run time performance but prolongs recovery time. A small log file degrades run time performance but reduces recovery time. A figure such as 32 blocks or 64 blocks is reasonable.

To configure your root (/) file system for **fsck** logging, specify the log size with the ROOTLOGSIZE parameter in the system file and rebuild the kernel. For example, the following line, added to your system file, sets the **fsck** log size for the root file system to 32 blocks:

```
ROOTLOGSIZE    32
```

The next time you boot the new kernel, **fsck** logging will be in effect for the root file system.

`Exportable`

This prompt appears when adding any type file system. If your system is not on a network, this feature does not apply to you.

Choose whether or not to export the file system, making it available to other systems on your network. Selecting this attribute adds the file system's name to the **/etc/exports** file.

If you choose to export the file system, the operation presents the Export Options query.

`Export Options`

This prompt appears when adding any type file system if you elect to make the file system exportable.

You may enter any of the following options, separating them with commas:

secure Require clients to use a more secure protocol when accessing the directory.

NOTE: Secure RPC using DES Authentication is an additional feature that must be purchased separately from the DG/UX ONC/NFS product. You must have this feature to use the **secure** option.

ro Export the directory read-only. If you do not specify the **rw** (read-write) option or the **ro** option, the directory is exported **rw**.

rw=*hostname*[**:***hostname*]...
Export the directory read-only to any systems except those specified in this option. Systems specified in this option have read-write access to the directory. Allowing another system read-write access to the directory does not override normal file and directory permissions. If you do not specify the **rw** option or the **ro** option, the directory is exported read-write to all.

anon=*uid*
If a request comes from an unknown user, use *uid* as the effective user ID. Superusers (UID 0) are considered **unknown** by the ONC/NFS server unless they are included in the **root** option below. The default value for this option is **–2**. Setting the value of **anon** to **–1** disables access by unknown users.

root=*hostname*[**:***hostname*]...
Give superuser access only to superusers from the specified hosts. By default, superusers from other systems do not have superuser access to the directory. A superuser is any user whose UID is 0 (**root** and **sysadm**, for example).

access=*client*[**:***client*]...
Give mount access to each client listed. A client can either be a host name or a net group (see the **netgroup**(4) manual page). Each client in the list is first checked for in the **/etc/netgroup** database and then the **/etc/hosts** database. The default value allows any system to mount the given directory.

After you have answered the queries above, the Add operation presents one more query before executing. If you made the file system exportable, the query is `Mount and export the file system`. If you did not make the file system exportable, the query is `Mount the file system`. Mounting the file system makes it immediately accessible on your system. Exporting the file system makes it immediately accessible to other systems on your network according to any export options you specified.

The Add operation then adds the file system entry to **fstab**. Then, if so requested, it attempts to mount the file system. If the mount directory does not already exist, it asks if it should create it for you before proceeding with the mount. If you chose to export the file system, the operation adds an entry to **/etc/exports** and calls **exportfs**(1M) to make the file system available to other systems in your network.

Deleting a Local File System

Select the Delete operation to remove one or more file systems from the file system table, **/etc/fstab**. If the selected file systems are exportable, the operation also removes their entries from the **/etc/exports** file.

The operation presents a list of file systems and lets you choose which ones to delete. You may also choose whether or not to unmount the file system after deleting it. If the operation cannot unmount the file system (because, for instance, the file system is in use), the operation displays a warning. The operation deletes the file system table entry even if it cannot unmount the file system.

Deleting a file system without unmounting it does not interrupt any work sessions currently using the file system. The next time mounting occurs (at boot or when coming up to run level

2), however, the file system will not be available. The data in the file system remains intact even if no entry for the file system exists in the **fstab** file. To make the file system available again, mount it.

You cannot unmount a file system while users are using it.

You cannot unmount a `ramdisk` type file system if it contains any files or directories. Unmounting a `ramdisk` type file system removes the associated physical device entry in **/dev**.

The operation asks for confirmation before deleting the file system entry.

Expanding a File System

To increase the size of a file system, select the Expand operation. This operation is also available in **diskman** in the File System Management Menu as Grow a File System.

Before you expand the root (/) or **/usr** file systems, remember that you cannot boot from a file system built on a multi-part logical disk. 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 diskman, **/usr/stand/diskman**, which you may need to boot to recover after a failure. To increase the size of either of these file systems, first rearrange the contents of the physical disk so that the desired free space is located immediately after the file system, thus allowing the Expand operation to stretch the file system's underlying single-piece logical disk rather than create a second piece.

You must unmount the file system before performing this operation. You may not use the Expand operation to increase the size of a striped logical disk. To change the size of a mirror, you must first break it apart so that its component images are available as individual logical disks. Expand the logical disks separately, then rebuild the mirror. When you rebuild the mirror, you will have to synchronize all but one of the images (the one you use as the master for synchronization).

The operation prompts you for:

- The name of the logical disk.
- The physical disk starting address for the new space.
- The size of the new space in 512–byte blocks. If the new space follows an existing piece of the logical disk, the operation simply extends the existing piece to provide the new space. If the new space does not follow an existing piece, the operation allocates the space as a new piece of the logical disk.

You may continue to add pieces to the logical disk until it has 32 pieces. Remember that you cannot boot from a logical disk that consists of more than one piece. Consequently, if expanding the **root** logical disk adds a new piece to it, you cannot boot the kernel, **/dgux**; if expanding the **usr** logical disk adds a new piece to it, you cannot boot stand–alone diskman, **/usr/stand/diskman**.

Keep in mind that file system internal data structures and the free space requirement (10% by default) will use some of the space that you are adding to the file system. Consequently, you should add 17% more space than the amount you intend to use. For example, if you need 30,000 more blocks of usable space in a file system, you should add 30,000 + (17% of 30,000) blocks, or 35,100 blocks.

The Expand operation maintains the required percentage of free space for the file system. By default, a file system has a free space requirement of 10%.

NOTE: Before beginning the operation, you should advise any remote hosts to unmount the file system. If they do not unmount it, they will not be able to access the file system or unmount it when it is available again after the operation. To restore access to the file system, the remote hosts will have to remount the file system at a different mount point directory. Before remounting it, the remote host may have to remove the file system's existing entry from the **/etc/mnttab** file.

Shrinking a File System

To decrease the size of a file system, select the Shrink operation. This operation is also available in **diskman** in the File System Management Menu as Shrink a File System.

You must unmount the file system before performing this operation. You may not use the Shrink operation to decrease the size of a striped logical disk. To change the size of a mirror, you must first break it apart so that its component images are available as individual logical disks. Shrink the logical disks separately, then rebuild the mirror. When you rebuild the mirror, you will have to synchronize all but one of the images (the one you use as the master for synchronization).

The operation prompts you for the number of 512–byte blocks that you want removed from the end of the file system. The operation selects which blocks to remove based on internal criteria.

The Shrink operation maintains the required percentage of free space for the file system. By default, a file system has a free space requirement of 10%.

Shrinking a file system requires the operation to rearrange data within the file system, collecting unallocated blocks so that the system may free the desired number. The internal structures of a file system complicate this process even more, particularly because some of these internal structures, like the file system itself, are designed to contain a certain amount of free space. The shrink operation will not transgress this built–in cushion of free space because doing so may result in poor I/O performance for the file system. If you specify that the shrink operation recover more space than is available from the file system, it returns an error.

To give you some flexibility, the shrink operation allows you to specify two file system parameters for the shrink operation:

`Percentage of blocks in the remaining DARs to keep unallocated`

You may set what minimum percentage of the file system should remain empty after the shrinking operation. This figure refers to the percentage of unallocated blocks per DAR (Disk Allocation Region). The default is 20%. If the shrink operation cannot free the desired number of blocks while leaving this percentage of remaining space free, the operation fails.

`Percentage of file node slots in the remaining DARs to keep available`

You may set what minimum percentage file node slots should remain available per

DAR after the shrinking operation. A DAR can hold a fixed number of files and directories (file nodes), and this percentage determines how many of them will remain available for directory or file creation. The default is 20%. If the shrink operation cannot free the desired number of blocks while leaving the desired percentage of file node slots available, the operation fails.

Under some conditions, the Shrink operation will reduce the file system more than the requested number of blocks. This happens when the Shrink operation causes the new end of the file system to fall in a DAR's metadata area (file node table or free space bitmap). The operation detects this condition and shrinks the file system further, to the beginning of the DAR. This condition does not constitute an error or a problem. When it occurs, the operation prints the following message:

```
The last DAR size was less than the minimal needed to support
the file node table and the DAR bitmap.  The new size of the
file system has been adjusted to nnnn blocks.
```

For a more detailed discussion of the file system internals, see the **tunefs**(1M) manual page.

Modifying a Local File System

To change the attributes associated with a local file system, select the Modify operation. The attributes you can change depend on the type of file system. For more information on the various attributes, see "Adding a Local File System Entry" earlier in the chapter.

You can modify a file system that is in use. After you have specified how you wish to modify the file system, the Modify operation asks if you want to apply the modifications immediately. If you choose to do so, the operation attempts to remount and then unexport or export (as appropriate) the file system so that the changes will be effective immediately.

Displaying Local File Systems

Select the List operation to display the current file system table (**/etc/fstab**) entries. You may choose to list all entries, only entries for mounted file systems, or only entries for exported file systems.

An example display follows:

```
                                                  FS            NFS   Dump  Fsck
File System Source        Mount Directory         Type       RW Mount Freq  Pass
------------------        ---------------         ----       -- ----- ----- ----
/dev/dsk/root             /                       dg/ux      rw         d    0
/dev/dsk/usr              /usr                    dg/ux      rw         d    0
/dev/dsk/usr_opt_x11      /usr/opt/X11            dg/ux      rw         d    1
/dev/dsk/usr_opt_aview    /usr/opt/aview          dg/ux      rw         d    1
/dev/ram2                 /ram_dir/ram2           dg/ux(ram) rw         x    0
```

The columns in the display correspond to the columns in the **fstab** file. The `File System Source` column shows the physical device pathname. The `Mount Directory` column shows the mount point directory. The `FS Type` column shows the file system type, where the values are:

`dg/ux`	A typical disk file system.
`dg/ux(ram)`	A `ramdisk` (memory) file system.

`cdrom`	The file system on a CD-ROM drive.
`dos`	An MS-DOS file system, created by **mkfs** with the **dos** flag.

The `RW` column gives the write permissions mode, either `rw` (read/write) or `ro` (read only). The `NFS Mount` column applies only to remote file systems mounted via ONC/NFS. The field is blank for a local file system. The `Dump Freq` column gives the backup frequency, where possible values are:

`d`	Backup during daily, weekly, and monthly backups.
`w`	Backup during weekly and monthly backups.
`m`	Backup during monthly backups.
`x`	Do not back up.

The `Fsck Pass` column shows during which pass the file system checker, **fsck** will check the file system. The value may be `0` through `9`.

For more information on the various **fstab** fields, see "Adding a Local File System Entry" earlier in the chapter.

Mounting and Unmounting a Local File System

After adding a file system with File System –> Local Filesys –> Add, you have to mount the file system to make it accessible to users. Mount a file system with the **sysadm** operation File System –> Local Filesys –> Mount. This operation calls the **admfilesystem**(1M) command, which invokes **mount**(1M).

Remote users to whom you have given access to the file system cannot mount it on their systems until you mount it locally on your own system.

When the Mount operation prompts you for the file system to mount, specify the mount point directory name. If there are no local file systems in your file system table that are currently unmounted, the operation reports this condition before terminating. If an attempt to mount a file system fails, it may be because the file system is corrupted. Verify the integrity of the file system with File System –> Local Filesys –> Check.

To make a file system inaccessible, unmount it with the **sysadm** operation File System –> Local Filesys –> Unmount. This operation calls the **admfilesystem**(1M) command, which invokes **umount**(1M).

You cannot unmount a file system that is in use. A file system is in use if it contains the working directory for any user or running process or if it contains a program that is currently running.

Exporting and Unexporting a Local File System

To make a local file system accessible to other systems on your network, export it with the **sysadm** operation File System –> Local Filesys –> Export.

You can export only file systems that you have made exportable either through the Add operation or the Modify operation. A file system is exportable if the **/etc/exports** file contains an entry for it. The Export operation calls the **exportfs**(1M) command to export the file system.

To make a local file system no longer accessible to other systems on your network, unexport it either with the **exportfs** command (with the **–u** option) or with the **sysadm** operation File System –> Local Filesys –> Unexport.

The Export and Unexport operations do not change the **/etc/exports** file, which is essentially a list of file systems for exporting. The Export and Unexport operations simply determine whether or not a file system will be available on remote systems at this time. If a file system appears in the **exports** file, the system makes it available every time the network software starts. The system-maintained file **/etc/xtab** contains a list of currently-exported file systems.

Checking a Local File System

To verify the consistency of a file system, use the **sysadm** operation File System –> Local Filesys –> Check.

The Check operation calls the file system checker, **fsck**(1M), to verify the internal structure of the file system. You can also check a file system by performing the Check a File System operation available in **diskman**'s File System Management Menu. See Chapter 7 for more information on **diskman**.

There are a number of options you can supply to the **fsck** command. The Check operation lets you specify any of these options. For more information on **fsck**, see "File System Checking: fsck," at the end of this chapter. Also see the manual page for **fsck**(1M).

You can only check unmounted file systems. When the system comes up, it will not mount file systems that are inconsistent and need to be checked. Typically, you only need to check file systems after a failure such as a power outage or disk head crash has caused the system to halt unexpectedly.

Managing Remote File Systems

The Remote Filesys menu provides operations for managing the remote file system entries in the file system table, **fstab**, and making remote file systems accessible on the local system.

To have access to a remote system's file systems, your system and the remote one must have the ONC/NFS network package installed and set up on your systems. For more information on ONC/NFS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

The following sections discuss the remote file system operations in more detail.

Adding a Remote File System

If you want users on your system to have access to a file system from another host on your network, add the remote file system with the operation File System –> Remote Filesys –> Add. This operation adds an entry to the file system table, **/etc/fstab**. The Add operation presents several queries, discussed below.

`Mount Directory`

This is the directory on your system where you want the file system to appear. If the directory does not already exist, the Add operation will ask if it should create the directory.

`Remote Host Name`

This is the host name of the system where the directory resides. This host name is the name by which the system is known on your network.

Systems connected to multiple networks have multiple network interfaces; therefore, they have multiple host names. Although you may be able to contact the system using any of the system's host names, it is more efficient (in terms of performance) to use the host name of the network interface connected to your network.

For example, you want to mount a file system that is located on a system known commonly as **sales03**. This system is connected to two networks and functions as the gateway between the two networks; therefore, the system has two network interfaces. One network interface is called **sales03**, and the other is called **sales03–alt**. Of these two possible host names, you should specify the one that is on the same network as your system. To determine an interface's network, look at its Internet address. See *Managing TCP/IP on the DG/UX™ System* for more information on networks.

`Remote Mount Directory`

This is the mount point directory where the file system is accessible on its native system. A file system must be mounted on its native system before other systems in the network can mount it.

`Write Permission`

If the write permission is `Read/Write`, local users may change the file system as well as read and execute files in it. You can mount the file system with `Read/Write` permission only if the remote host has exported the file system in **rw** mode. The `Read/Write` mode does not allow users to override the normal file system security architecture. If the write permission is `Read Only`, local users may only read and execute files in the file system.

`NFS Mount Type`

If an ONC/NFS file system is hard mounted, user processes will wait indefinitely for file system accesses to complete. This means that if the remote system on which the file system resides is not responding (because it is down, for example), user programs will appear to hang while they wait for the remote system to become accessible again. You cannot interrupt this kind of hard mount hang unless you specify the **intr** option when mounting the file system. Read the information on **intr** in the **mount**(1M) man page before using it.

The benefit of the hard mount option is that it provides more reliability when writing to a remote file system. Hard mounting is preferable for file systems containing valuable data that you update from your remote system.

If an ONC/NFS file system is soft mounted, user processes will time out (terminate with an I/O error) if the remote file system does not respond within a reasonable period of time. The benefit of a soft mount is that the user process does not hang waiting for the remote file system to complete the requested disk I/O. The disadvantage is that there is some risk that a write operation will fail and cause you to lose the data you intended to write. Soft mounting is preferable for read-only file systems or for file systems containing data that you can afford to lose because of a failed write.

For a more complete discussion of the mount options, see the **mount**(1M) manual page or see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

`Interruptible`

Set this attribute to allow you to interrupt remote file system access when it appears hung or becomes inconveniently slow. You interrupt file access the same way you interrupt any other process. For example, the default shell **stty** settings allow you to interrupt a command with **^C**. If you executed the **ls** command in an interruptible, hard–mounted file system whose server had crashed, you would be able to interrupt the **ls** command with **^C**. If the file system were hard-mounted uninterruptible, however, you would not be able to interrupt the **ls** command, and your shell process would hang until the remote server restored access to the file system.

In addition to the interrupt that you can generate using the appropriate control–key sequence from the shell, interrupts may also result when the **kill**(1) command or **kill**(2) system call send a signal to a process or when a modem detects a hangup condition.

The `Interruptible` attribute is more useful for hard–mounted file systems because accesses to hard–mounted file systems will retry indefinitely until they succeed. In the case of soft–mounted file systems, on the other hand, a failed access will only retry until a finite (and usually brief) time–out period has elapsed.

`Retry in background`

If the remote host does not respond to your system's request to mount the file system, your system retries the mount request a number of times. If attempting a hard mount, your system will continue to retry the mount until the remote host responds. If attempting a soft mount, your system will retry the mount only a few more times.

This query allows you to choose whether the retry occurs in the background or in the foreground. This characteristic is significant when the system is booting because it determines whether the system may continue booting while a mount attempt retries. By putting mount retries in the background, the system may continue booting (mounting other file systems, starting services, and so on) without waiting on the mount retries.

For more information on how the **mount** command handles failures and retries, see the **mount**(1M) manual page or *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

After you have answered the queries above, the Add operation presents the prompt, `Mount the file system`. If you elect to mount the file system, the operation attempts the mount after adding the file system entry to **fstab**. If you select the **soft** ONC/NFS mount type for a read/write file system, the operation presents a confirmation box verifying that you really want the file system soft mounted.

After the Add operation adds the file system entry to **fstab**, it attempts to mount the file system if so requested. If the mount directory does not already exist, it asks if it should create it for you before attempting the mount.

Deleting a Remote File System

Select the Delete operation to remove one or more remote file systems from the file system table, **/etc/fstab**.

The operation presents a list of file systems and lets you choose which ones to delete. You may also unmount the file systems after deleting them. If an attempt to unmount a file system fails (because, for instance, the file system is in use), the operation displays a warning. The operation deletes the file system table entry even if it cannot unmount the file system.

Removing the file system entry does not interrupt any work sessions currently using the file system. The next time mounting occurs (at boot or when coming up to run level 3), however, the file system will not be available.

Modifying a Remote File System

To change the attributes associated with a remote file system, select the Modify operation. For more information on the various attributes, see "Adding a Remote File System Entry" earlier in the chapter.

You can modify a file system that is in use. After you have specified how you wish to modify the file system, the Modify operation asks if you want to apply the modifications immediately. If you choose to do so, the operation attempts to remount the file system if it is mounted.

Displaying Remote File Systems

Select the List operation to display the current file system table (**/etc/fstab**) entries. You may choose to list all entries or only entries for mounted file systems.

An example display follows:

```
                                                FS          NFS   Dump  Fsck
File System Source       Mount Directory        Type     RW Mount Freq  Pass
------------------       ---------------        ----     -- ----- ----- ----
sales03:/pdd/sales03     /pdd/sales03           nfs      rw hard  x     0
sales03:/sales/accounts  /sales/accounts        nfs      rw hard  x     0
sales01:/udd/sales01     /udd/sales01           nfs      ro soft  x     0
```

The columns in the display correspond to the columns in the **fstab** file:

`File System Source`	The mount directory name on the remote host where the file system resides.
`Mount Directory`	The mount point directory on the local system.
`FS Type`	The file system type, where the value for a remote file system is always `nfs`.
`RW`	The write permissions mode, either `rw` (read/write) or `ro` (read only).
`NFS Mount`	How the file system is mounted, `hard` or `soft`.
`Dump Freq`	The backup frequency. For a remote file system, this field should always be x because you only back up local file systems.
`Fsck Pass`	The pass in which the file system checker, **fsck**, will check the file system. For remote file systems, this value should always be 0 because you only check local file systems.

For more information on the various **fstab** fields, see "Adding a Remote File System Entry" earlier in the chapter.

Mounting and Unmounting a Remote File System

After adding an entry for a file system to the file system table (**fstab**), you have to mount the file system to make it accessible to users. You can mount a file system either with the **mount**(1M) command or with the operation File System –> Remote Filesys –> Mount.

When the Mount operation prompts you for the file system to mount, specify the mount point directory name, the second field in the **fstab** entry. If there are no remote file systems in your file system table that are currently unmounted, the operation reports this condition before terminating.

If your system's attempt to communicate with the remote system fails, your system will retry the mount request a number of times, possibly causing the Mount operation to appear to hang. For more information on the **mount** command and how it handles request retries, see **mount**(1M). You cannot mount a remote file system unless it is mounted on the remote system.

To make a file system inaccessible, you unmount it. You can unmount a file system either with the **umount**(1M) command or with the operation File System –> Remote Filesys –> Unmount.

You cannot unmount a file system that is in use. A file system is in use if it contains the working directory for any user or running process or if it contains a program that is currently running.

Managing the Swap Area

A swap area is not a file system, but the DG/UX system manages 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 logical disks as swap areas. The Swap Area menu provides operations for managing swap area entries in **fstab**.

Adding Swap Area

By default, your system has one swap area 24 MB (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.

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 MB of memory should have 24 MB (50,000 blocks) of swap area.

You know you have too little swap area when processes terminate unexpectedly and messages like this appear on your system console:

```
From system:  Out of paging area space
```

A system may have swap area on its own disks, or it may have swap area on another system accessible over the network. You may not have swap area 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 a server on the network. For more information on the local swap/remote OS configuration, see *Customizing the DG/UX*™ *System.*

To add local swap area in addition to the default swap area already established for your system, use the Logical Disk Management Menu of **diskman** to create a logical disk of the desired size. You do not need to create a file system on a logical disk to be used for swap area. See Chapter 7 for more information on **diskman**.

After you have created the logical disk, add the swap area to **fstab** and make it usable on the system by selecting the operation File System –> Swap Area –> Add.

The Add operation lists the available logical disks, the ones not already appearing in **fstab**, and lets you choose one. It then adds the **fstab** entry and calls **swapon**(1M) 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**(1M) program, when executed by the system at boot time, will run with the **–a** argument. Make this change by selecting the operation System –> Parameters –> Set. In the **Arguments to swapon** query, specify **–a**. The **–a** option to **swapon** causes it to mount all swap areas listed in **fstab** at boot.

Deleting Swap Area

To remove a swap area entry from the **fstab** file, select the 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 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 logical disks entered in the file system table as swap area.

Expert File System Information

Information in this section is optional. You do not need to read this section to manage your file systems; however, you may find this information useful if you have to diagnose problems with file systems.

File System Checking: fsck

The **fsck**(1) program checks and repairs file systems. The **fsck** program can check file systems in either of two modes, depending on the kind of file system:

`Multi-pass fsck`
: Checks normal file systems by performing five passes through the file system, each pass verifying different aspects of the file system.

`Fast recovery fsck`
: Performs single-pass verification of file systems that were mounted to take advantage of the **fsck** logging feature, described in the next section.

You may invoke **fsck** three different ways. In **diskman**, go to the File System Management Menu and select the Check a File System operation. In **sysadm**, select the operation File System –> Local Filesys –> Check. The third way is to invoke **fsck** directly at the shell prompt. For more information, see "Invoking the fsck Program" later in the chapter.

You need to check a file system only after a system crash, disk or disk controller hardware failure, or any other time when an abnormal event may have caused service to the file system to terminate unexpectedly. Normally, the system checks file systems as necessary when it boots.

You must use this program when you are bringing up your system after an abnormal shutdown such as a power outage or system crash.

The **fsck** program checks blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region (DAR) information, including the free-block bitmap, the free-inode list, and summary counts. Multi-pass **fsck** checks these features in this order. Fast recovery **fsck** does not necessarily check them in this order. Both modes of **fsck** report inconsistencies that they find. It is your option to fix the inconsistencies or ignore them.

For more information on the errors and messages that **fsck** returns, see Appendix B.

This section:

- Discusses the normal updating of the file system.
- Discusses the possible causes of file system corruption.
- Presents the corrective actions taken by **fsck**. It describes both the program and the interaction between the program and the system administrator.

Fast Recovery File Systems

You can reduce the amount of time that **fsck** requires to check a file system by requesting **fsck** logging for the file system. For fast recovery file systems, the system keeps a log containing records about modifications to the file system. If a failure occurs and you need to restore the file system, **fsck** can use the log to speed the process of verifying and restoring file system integrity.

Logging is good for file systems where it is crucially important to minimize the amount of time during which the file system is unavailable (as during verification and repair). Because logging has some negative impact on run time write performance in the file system, we recommend it primarily for file systems where rapid recovery and high availability are crucial.

Any file system can be a fast recovery file system. You specify **fsck** logging for a file system when you mount it. Mount a file system with the **sysadm** operation File System –> Local Filesys –> Mount.

Select the **Fsck Logging** feature when prompted. The operation will later prompt you for the log size. If you edit the **/etc/fstab** file yourself, you add the **fsck_log_size** option in the options field, specifying a size for the **fsck** log as in the following sample **fstab** entry:

```
/dev/dsk/sales /sales dg/ux rw,fsck_log_size=64 d 1
```

You specify the log size in 512-byte blocks. There is a tradeoff in performance between log files of different sizes. A large log file improves run time performance but prolongs recovery time. A small log file reduces recovery time but degrades run time performance.

To configure your root (/) file system for **fsck** logging, specify the log size with the ROOTLOGSIZE parameter in the system file and rebuild the kernel. For example, the following line, added to your system file, sets the **fsck** log size for the root file system to 32 blocks:

```
ROOTLOGSIZE    32
```

The next time you boot the new kernel, **fsck** logging will be in effect for the root file system.

File System Update

Every time a file is modified, the DG/UX operating system performs a series of file system updates. When written to disk, these updates yield a consistent file system.

There are five types of file system updates. The updates involve the following areas:

- the superblock
- inodes
- index (indirect) blocks
- data blocks (directories and other files)
- disk allocation region information, which includes the free-block bitmap and the inode table

Corrupted File Systems

Many things can corrupt a file system. Improper shutdown procedures and hardware failures are the most common causes.

Some examples of improper shutdown procedures are:

- Forgetting to use the **shutdown**(1M) command (which unmounts all file systems, including the root) before halting the CPU.
- Physically write protecting a mounted file system.
- Taking a mounted file system off line.

Each DG/UX file system contains a flag in the superblock which indicates whether or not the file system is mountable. You can only mount a file system if it is marked as mountable; if a file system is unmountable, you must first run **fsck** in order to repair inconsistencies. The **fsck** program will mark a file system mountable only when it is internally consistent. If an attempt to mount a file system fails with the message `No space left on device`, the file system is probably corrupt and needs to be checked.

A file system is marked mountable when it is created. It is marked unmountable whenever it is mounted, and is not marked mountable again until it is either unmounted or has passed the **fsck** program's internal consistency checks. Therefore, file systems which were still mounted at the time of an abnormal system shutdown cannot be remounted until **fsck** has been run over them, whereas those file systems which were cleanly unmounted before shutdown can immediately be remounted.

Fixing Corrupted Files

This section discusses ways to discover and fix inconsistencies for different kinds of update requests.

The **fsck** program lets you check a file system for structural integrity by performing consistency checks on redundant data. Redundant data is either read from the file system or computed from other known values. When **fsck** reports an inconsistency, it asks whether the inconsistency is to be corrected by repairing or deleting the corrupted item. You can accept or reject this request. In the following example, **fsck** finds an incorrect link count in Phase 4 and asks if it should fix the problem. The file system being checked was not mounted for fast recovery.

```
# fsck /dev/dsk/mydisk ↵

** /dev/dsk/mydisk:
** Phase 1 - Check Blocks and File Sizes
** Phase 2 - Check Directory Contents
** Phase 3 - Check Connectivity
** Phase 4 - Check Link Counts and Resource Accounting

Inode 67 (owner: 2 [bin]; group: 2 [bin]; size: 52736 bytes;
    type: Ordinary; mode: 755; mtime: Fri Nov 20 17:54:36 1987)
    has incorrect link count (2 should be 1) -- fix?  y ↵

** Phase 5 - Check Disk Allocation Region Information
File system is now mountable.

13936 of 50000 blocks used (36064 free); 288 of 5822 inodes
    used (5534 free).

#
```

If you respond **y**, the **fsck** program corrects the incorrect link count that it found for inode 67 and moves on to Phase 5. When it finishes, it reports that the file system is mountable.

If you respond **n**, **fsck** does not fix the inconsistency, and the file system remains unmountable.

Superblock and Disk Allocation Region Information

The superblock and disk allocation region information are some of the most commonly corrupted items. Every change to the file system's blocks or inodes modifies the superblock and the disk allocation region information. The superblock and disk allocation region are most often corrupted when the system was not properly shut down with the **shutdown**(1M) command.

Superblock and disk allocation region inconsistencies can involve file system size, the number of available inodes and blocks, the free-block bitmaps and the free inode lists. The following sections give brief discussions of these sections.

Free-Block Bitmap
Each disk allocation region (DAR) contains a bitmap representing all the blocks in the DAR. Multi-pass **fsck** compares that information with its own map of allocated blocks, looking for inconsistencies that suggest DAR corruption.

Free-Inode List
Each DAR contains a link list of free inodes in that DAR. The **fsck** program ensures that all free inodes in the DAR appear in the list, and that no allocated inodes appear in the list.

Summary Counts
The superblock and each DAR contain several counts: the number of used inodes, the number of used blocks, the number of directories. The **fsck** program compares these counts to the information it has compiled.

Inodes

An individual inode is less likely than the superblock to be corrupted. However, because of the great number of active inodes, the free inode lists are as susceptible as the superblock to corruption. Multi-pass **fsck** checks for inconsistencies involving format and type, link count, duplicate blocks, and inode size. Fast recovery **fsck** checks for inconsistencies involving link count, duplicate data element pointers, and inode size.

Format and Type
Each inode contains mode information. This information describes the type of the inode. Inodes may be one of eight types: regular, directory, control point directory, special block, symbolic link, special character, FIFO, or socket. Any other type is illegal.

Link Count
Each inode contains a count of the directory entries linked to the inode. Multi-pass **fsck** verifies the inode count by checking down the total directory structure, starting from the root directory, and calculating an actual link count for each inode. Fast recovery **fsck** verifies link counts by comparing them to link count records in the log.

In multi-pass checking, when the link count (which is stored on the disk) is nonzero and the actual link count (kept by **fsck**) is zero, no directory entry appears for the inode. If no entry appears, **fsck** may link the disconnected file to the **lost+found** directory. Fast recovery has records that enable it to match lost files with their original directories; consequently, it does not leave files or file fragments in **lost+found**.

If the stored and actual link counts are nonzero and unequal, **fsck** may replace the link count on the disk by the actual link count. When this situation arises, a directory entry may have been added or removed without the inode being updated.

Duplicate Blocks
Each inode contains a list and sometimes pointers to lists (index blocks) of all the blocks claimed by the inode.

Multi-pass **fsck** checks these lists for duplicate blocks. Duplicate blocks can occur when a file system uses blocks claimed by both the free-block bitmap and other parts of the system or when two or more inodes claim the same block. Any block claimed more than once is flagged by **fsck** as a duplicate block. If there are any duplicate blocks, **fsck** makes a partial second pass of the inode list to find the inode of the duplicated block. If the files associated with these inodes are not examined for correct content, **fsck** will not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modification time is incorrect and should be cleared.

Fast recovery **fsck** does not check for duplicate blocks. It compares inodes and index blocks with log records listing which blocks were allocated for each file.

Size Checks
Each inode contains a size field. This field's size indicates the number of bytes in the file associated with the inode.

The **fsck** program can check the size for inconsistencies, such as directory sizes that are not a multiple of 512 bytes, or a mismatch between the number of blocks actually used and the number indicated by the inode size. The **fsck** program also checks for directory corruption, where conflicting information is found within the directory entries.

The **fsck** program can also perform a check of the size field of an inode. Multi-pass **fsck** uses the size field to compute the number of blocks that should be associated with the inode, and then compares that number to the actual number of blocks claimed by the inode. Fast recovery uses log records to track changes to the file size and number of blocks claimed by the inode.

Control Point Directories
The root inode of a file system is a special type of directory known as a *control point directory.* A control point directory is like an ordinary directory except that it has resource limits associated with it for inodes and for data blocks. Effectively, these limits determine how many files you may create in the file system and how large the file system (total size of all files and directories) may become. The total resources consumed by the control point directory and all its descendants (to which it is the *space parent*) may not exceed the size limits. The limit on data blocks (size) is determined by the size of the logical disk on which the file system resides. To change the size of the file system, use the Expand and Shrink operations in the File System -> Local Filesys menu.

Index Blocks

Index blocks (also known as indirect blocks) are owned by an inode. Therefore, inconsistencies in an index block directly affect the inode that owns the block.

Multi-pass **fsck** can check inconsistencies involving blocks already claimed by another inode and block numbers outside the range of the file system.

Fast recovery **fsck** uses the log to track changes to the contents of index blocks.

Data Blocks

There are two types of data blocks: plain data blocks and directory data blocks. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries. The **fsck** program does not try to check a plain data block.

There are several checks that **fsck** makes for directory data blocks. Multi-pass **fsck** checks all entries in all directory data blocks, but fast recovery **fsck** checks only those entries that have been modified. The directory data block check searches for:

- bad self-identification information
- directory entries for unallocated inodes
- directory entries for inodes which do not exist in the file system
- directories that are disconnected from the file system

If a directory entry inode number points to an unallocated inode, **fsck** may remove that directory entry. Directory entry inode numbers can point to unallocated inodes when the data blocks containing the directory entries are modified and written out, but the inode is not written out.

If a directory entry inode number is pointing to a nonexistent inode, **fsck** may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The **fsck** program checks that all directories are linked into the file system; i.e., they have a parent directory pointing to them (except for the root). Multi-pass **fsck** links unlinked directories to the file system's **lost+found** directory. Fast recovery **fsck** links unlinked directories back to their original places in the directory tree. When inodes are being written to the file system without the corresponding directory data blocks being written to the file system, the directories are not linked into the file system.

Invoking fsck

You can invoke **fsck** any of five ways:

sysadm Select the operation File System –> Local Filesys –> Check.

rc script You can set up any of your system's **rc** scripts to invoke **fsck** when you change run levels. For more information on **rc** scripts and how to set them up, see Chapter 3.

Command line From the command line, type:

fsck *[options] [file_system_names]*

where *options* are single character flags that modify the behavior of the command and *file_system_names* refer to the file systems that you want to check. The *options* are covered in detail later in this section.

Initialization The initialization version of **fsck** is built into the operating system kernel and is automatically run over the root file system when you boot your system.

Stand-alone You boot stand-alone **diskman**, go to the File System Management Menu, and select the Check a File System operation. See Chapter 7 for information on **diskman**.

Options to fsck

All options are represented by single-character flags; options must begin with a hyphen. All options except for **–t** are Boolean flags, and may thus be combined. For example, you can combine the options **–p**, **–x**, and **–D** as follows: **fsck –pxD**.

The following options are interpreted by **fsck**:

–l Perform fast recovery using the **fsck** log, if possible. The **fsck** program can perform fast recovery only if:

- You specified the **fsck_log_size** option, which turns on fast recovery logging, the last time you mounted the file system (unless it was the root file system, for which fast recovery logging is in effect if you set the ROOTLOGSIZE parameter when you built the kernel), *and*
- This is the first time that **fsck** has checked the file system since that mount.

If fast recovery **fsck** cannot repair the file system, or if you specify the **–l** option for a normal-recovery file system, **fsck** performs a multi-pass check.

–p Detect all possible inconsistencies, but correct only those inconsistencies that may be expected to occur from an abnormal system halt. For each corrected inconsistency, one or more lines will be printed identifying the file system and the nature of the correction. Any other inconsistencies will cause the check of that file system to fail. The following 15 inconsistencies (and only those listed) will be corrected for the specified file systems:

1. An inode has an incorrect count of the blocks it uses. The count is corrected.
2. An inode is partially truncated. Partial truncation can occur if the system is abnormally halted while a file is being truncated, leaving the file claiming more data blocks than its size in bytes would require. The extra blocks are freed.
3. A directory has an incorrect child count. The count is corrected.
4. A directory entry exists for an inode which is unallocated. The directory entry is removed.
5. A directory entry's file name length is incorrect. The length is corrected.
6. An inode is unreferenced (has no directory entries anywhere in the file system). The inode is reconnected in the **lost+found** directory.
7. No **lost+found** directory exists, but an inode needs to be reconnected there. The **lost+found** directory is created.
8. The root directory needs to be expanded in order to make room for a **lost+found** directory entry. The directory is expanded.
9. The **lost+found** directory needs to be expanded in order to make room for a directory entry for an inode being reconnected there. The directory is expanded.
10. An inode's link count is incorrect. The count is corrected.
11. The root control point directory's resource accounting (blocks, inodes) is incorrect. The counts are corrected.
12. A disk allocation region (DAR) has an incorrect free-block bitmap. The bitmap is corrected.
13. A DAR has an incorrect free-inode list. The list is corrected.

14. A DAR has incorrect summary counts of used blocks, inodes or directories. The counts are corrected.

15. The summary counts in the superblock are incorrect. The counts are corrected.

–q Repair the inconsistencies listed under the **–p** option automatically, without asking for user approval. Unlike **–p** however, more serious inconsistencies will not cause **fsck** to fail; the user must still answer the resulting queries.

–y Audit and interactively repair all file system inconsistencies assuming a "yes" response to all questions asked by **fsck**.

CAUTION: Use this option with great care, since it could lead to irreversible changes to the file system.

–n Audit and interactively repair all file system inconsistencies, assuming a "no" response to all questions asked by **fsck**. This option also means that all file systems will be opened with read-only intent.

–x Look at a file system's superblock to see if it is marked mountable. If so, do not check the file system for inconsistencies. If the file system is marked unmountable, check it.

–s Ignore the actual free-block bitmap and unconditionally reconstruct a new one.

–S Conditionally reconstruct the free-block bitmap. A free-block bitmap is reconstructed if and only if the file system is consistent. This option also forces a "no" response to all **fsck** questions.

–D Directories are checked for bad blocks.

–f Fast check: blocks and sizes are checked; the free block bitmap is reconstructed if necessary.

The following options are mutually exclusive, and use of more than one per invocation is not allowed: **–l**, **–y**, **–n**, **–p**, **–q**, and **–S**.

Arguments to fsck Options

If you do not specify which file systems to check on the **fsck** command line, **fsck** will check those having appropriate entries in the **fstab** file. An appropriate entry is one having a nonzero pass number and a "rw" or "ro" mounting status. If the **–p** option was specified, the checking occurs in order of pass number, with those file systems of equal pass number being checked in parallel with each other. Otherwise, checking occurs in order of appearance in **fstab**.

If arguments are specified (the rest of the command line after the option flags), those file systems, and only those file systems, are checked sequentially in the order given.

File systems may be specified as arguments to **fsck** in one of two ways: by the special device file (in **/dev/dsk** or **/dev/rdsk**) containing the file system; or by the directory that **/etc/fstab** indicates will serve as the mount point for the file system.

Checking File Systems

File system checking proceeds without any input from the operator if no errors are discovered. When a fatal inconsistency is discovered, no further checking is done on that file system; the

fsck program either exits or proceeds to the next specified file system. When an inconsistency is discovered with the **–p** option, and that error is one of those listed under **–p**, the inconsistency is fixed without operator intervention. Any other discoveries of inconsistencies require the operator to make a decision. The **fsck** program prompts with its recommended action. If you answer **yes**, **fsck** takes the recommended action. In no case will any damaging action be taken without approval. Note, however, that advance approval or disapproval may be given by invoking **fsck** with the **–y** and **–n** options, respectively.

The **fsck** program will refuse to check any file system for which any of the following conditions hold true:

- The file system is mounted (except when you specified the **–n** option, which opens the file system read-only).
- The special file associated with the file system cannot be opened.
- The specified pathname (or its device node associate in **/etc/fstab**) is not a block-special, character-special, or regular file whose size can be determined.

The **fsck** program checks for the following inconsistencies (the term "Bad format" refers to system blocks which do not have the required self-identification information).

- Unreadable or inconsistent superblocks.
- Bad format in superblocks.
- Invalid contents in superblock's reserved area.
- Bad value for superblock's file system size.
- Bad value for superblock's DAR size.
- Bad value for superblock's inode/DAR density.
- Bad value for superblock's default data element size.
- Bad value for superblock's default index element size.
- Bad value for superblock's default directory data element size.
- Bad value for superblock's default directory index element size.
- Bad value for superblock's default first allocation threshold.
- Bad value for superblock's default second allocation threshold.
- Bad format in inode table block.
- Invalid contents in inode's reserved area.
- Files of unknown type.
- Files with bad fragment size.

- Files which are partially truncated.
- Files claiming impossible blocks.
- Files claiming system-area blocks.
- Bad Index-block format.
- Files with incorrect block counts.
- Files claiming already-claimed blocks.
- Unallocated root inode.
- Bad file type for root.
- Incorrect resource limit information in root.
- Incorrect parent directory in root.
- Directories with "holes" (unallocated blocks before end-of-file).
- Bad format in directory blocks.
- Directories with invalid information in reserved areas.
- Directories with empty blocks at end.
- Directories with incorrect child counts.
- Extra directory entries named "." or ".."
- Directory entries with invalid characters in file names: "/" or non-ASCII characters.
- Directory entries which would have too-long pathnames.
- Directory entries which are out of order.
- Directory entries with incorrect entry lengths.
- Directory entries with incorrect file name lengths.
- Extraneous hard links to directories (including cycles in file system name space).
- Extraneous hard links to Symbolic Link files.
- Directory entries to invalid inodes.
- Directory entries to unallocated inodes.
- Files with incorrect space parent.
- Unconnected files or directories.

Licensed Material – Property of copyright holder(s)

- Bad or missing **lost+found** directories.
- Bad **lost+found** directory entries.
- Root or **lost+found** directories needing expansion.
- Files with incorrect link counts.
- Incorrect resource allocation counts in control point directories.
- Bad format in DAR blocks.
- Invalid contents in reserved area of DAR blocks.
- Incorrect free-block bitmaps in DARs.
- Incorrect or incomplete free-inode lists in DARs.
- Incorrect DAR summary counts: blocks used, inodes used, directories used.
- Incorrect superblock summary counts.

fsck Output

See Appendix B for a complete list of **fsck** errors.

If the **–p** option is used, **fsck** prints out one or more lines for each inconsistency it corrects, indicating the file system fixed and the error corrected. After successfully checking or correcting a file system, **fsck** prints out the name of the file system, the number of files on it, and the number of free and used blocks.

If you do not specify **–p**, **fsck** is more verbose. It will first print out the name of the file system. If performing a multi-pass check, **fsck** prints a message as it enters each phase of checking a file system. A message is printed for each inconsistency encountered, and the operator is prompted for approval before each correction is attempted. (If the **–y** or **–n** flags are used, **fsck** automatically answers such prompts itself.) When checking is complete for the file system, a message is printed if any corrections were made. Finally, the numbers (used, free and total) of files and blocks are printed.

The **fsck** program attempts to give as much information as possible about any files for which you must make decisions (such as whether to remove it, etc.). At least the following information will always be displayed:

- I-number, which is a number specifying a particular inode on a file system
- Owner's user ID
- Owner's group ID
- File type
- Mode

- Size
- Time of last modification

When possible, the following additional information will be displayed:

- Pathname
- Owner's username
- Owner's group name

End of Chapter

Chapter 9
Managing Terminals and Ports

This chapter tells how to manage your system's ports, including terminal lines and the lines you use for UUCP connections. The DG/UX system provides port services through the Service Access Facility (SAF).

For more information on setting up terminals and keyboards to accommodate your environment, see *Customizing the DG/UX™ System.*

You can manage port services two ways. One way is by using the operations found in **sysadm**'s Port menu, located in the Device menu. The first section of this chapter, "Terminal and Port Operations," covers these operations.

The second way to manage ports is with commands invoked at the shell level. For coverage of the shell commands, see the second part of the chapter, "Expert Information."

Briefly, SAF is controlled by a single process on your system, **sac**, the service access controller. 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. Figure 9–1 shows the SAF process structure.

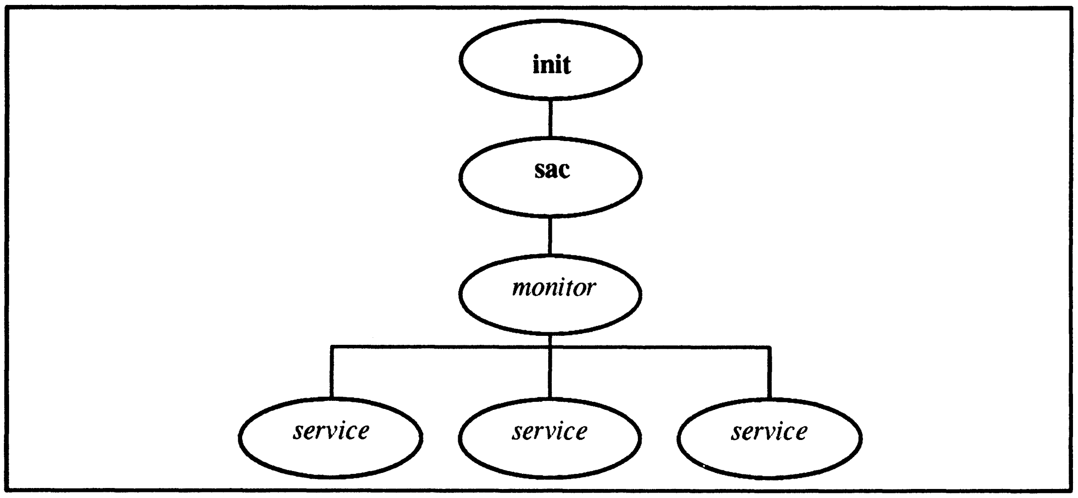

Figure 9–1 SAF Process Structure

For detailed treatment of the Service Access Facility, see the second part of the chapter, "Expert Information."

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. These operations are actually just streamlined versions of the operations in the Port Services menu, offered 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.

Managing Terminals

Although you may manage user terminals (also called TTYs) using the operations in the Port Monitor and Port Service menus, you may find it easier to use the operations in the Terminal menu instead. The operations perform essentially the same functions, but the Terminal menu operations are tailored specifically for managing user terminals.

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 one does not, the operation adds and starts a **ttymon** port monitor. The port monitor, whose tag (name) is **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. 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.

If you intend to support more than 128 terminals on your system, you should consider creating additional port monitors. 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 a maximum of around 128 terminals per port monitor. To add a port monitor, use the operation Device –> Port –> Port Monitor –> Add.

The Terminal –> Add and Terminal –> Modify operations 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 tag (name) of the port monitor to which this 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 multiple **ttymon** port monitors on your system, the operation prompts you for the tag 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.

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 above, “Adding and Modifying Terminals,” for a discussion of these attributes.

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.

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. This port monitor replaces the now–obsolete **getty** and **uugetty** programs. Generally, you should have one **ttymon** port monitor for every 128 terminal lines. 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 more information on 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.

`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 reference manual page for **doconfig**(3N).

`Comment`

This comment may be any text that you wish to add to describe the port monitor for your own records.

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. 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     #
```

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

Adding and Modifying 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. 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.

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.

Queries for Adding or Modifying ttymon and listen Services

`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 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.

Queries for Adding or Modifying 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 bidirectional 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 bidirectional 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 bidirectional use.

`Wait-read`
: Select this option if you want the port monitor to delay sending out a prompt on the line until it has received a particular number of New Line characters.

`Wait-read count`
: If you selected the wait-read option in the previous query, 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 did not select the wait-read option, the wait-read count has no effect.

`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 more 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.

Queries for Adding or Modifying 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

Use the List operation to display port service definitions. The operation displays definitions for all port services. The display is generated by **pmadm –l**, discussed in "Expert Information."

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.

Expert Information

This section provides technical detail about SAF (Service Access Facility). The preceding portion of the chapter tells how to use the **sysadm** operations to manage port services; this section describes the underlying functionality.

The Service Access Facility generalizes 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. As part of this simplification, the port monitor **ttymon** has replaced **getty** and **uugetty** for local access to login service although **ttymon** is not limited to login access.

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 more information on TLI, see *Programming with TCP/IP on the DG/UX™ System.*

The "Expert Information" section is divided into six subsections. The first, "Overview of the Service Access Facility," describes

- the Service Access Facility (SAF),
- its directory structure,
- its controlling program (the Service Access Controller or SAC),
- the configuration files that may be used to change the environment under which SAC, port monitors, and services operate, and
- the SAF's two administrative files.

The second section, "Port Monitor Management," describes the SAC administrative command **sacadm** and how it is used to manage the information in the SAC administrative file. This includes:

- printing port monitor status information
- adding a port monitor to the system
- enabling or disabling a port monitor
- starting or stopping a port monitor, and
- removing a port monitor from the system.

This section also describes the per–system and per–port monitor configuration scripts used to modify the SAC and port monitor environments and shows how to print, install, and modify each type of script.

A summary of the commands used to administer a port monitor is included at the end of the section.

The third section, "Service Management," follows the same outline. It describes the port monitor administrative command, **pmadm**. **pmadm** manages the information in the port monitors' administrative files. These files, one for each port monitor, include administrative and state information for each port and information about the service each port invokes. The **pmadm** command allows the system administrator to

- print information derived from the administrative file
- add and remove services
- enable and disable services
- print, install, and change per–service configuration scripts

A command summary is included at the end.

The fourth section, "The Port Monitor ttymon" describes what **ttymon** does and shows how to perform some of the administrative tasks described in the sections on port monitor management and service management—this time from the point of view of **ttymon**. These tasks include:

- listing the **ttymon** port monitors that have been configured
- listing the services that have been configured under a given **ttymon** port monitor
- enabling and disabling **ttymon** ports and services
- adding and removing a **ttymon** port monitor
- adding a service under a **ttymon** port monitor

The fifth section, "Terminal Line Settings," covers the **ttydefs** file, which replaces **gettydefs** as the database file for system TTY terminal information. The **ttymon** port monitor uses the file and the system administrator modifies it in the process of **ttymon** administration. Setting terminal modes and line speeds includes:

- printing terminal line setting information

- modifying terminal line settings
- setting up hunt sequences
- adding and removing terminal line settings for a terminal
- setting terminal options available using the **stty** command

A summary of commands used in **ttymon** and terminal line settings administration is included at the end of the section.

The sixth section, "The Port Monitor listen," describes what **listen** does and how you perform **listen**–related administrative tasks. These include:

- listing configured **listen** port monitors
- listing services available through a given **listen** port monitor
- enabling and disabling **listen** ports and services
- adding and removing **listen** port monitors
- adding a **listen** service

Tasks associated with port and service administration may be performed using either the operations in the **sysadm** utility or shell commands entered on the command line.

The shell commands for managing port monitors and port services are described in "Port Monitor Management" and "Service Management."

Overview of the Service Access Facility

The Service Access Facility (SAF) provides a mechanism for uniform access to services. From the system administrator's point of view, 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 the Service Access Facility, a service is a process that is started. There are no restrictions on the functions a service may provide.

The Service Access Facility consists of a controlling process, the Service Access Controller(SAC), and two administrative levels corresponding to two levels in the supporting directory structure. The top administrative level is concerned with port monitor administration, the lower level with service administration.

From an administrative point of view, the Service Access Facility consists of the following components, each of which is described in this section:

- The Service Access Controller

- 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 files 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 is the overseer of the server system. It is the Service Access Facility'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 specified by the system administrator. These states include: STARTING, ENABLED, DISABLED, STOPPING, NOTRUNNING, 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. Briefly:

- It customizes its own environment.
- It starts the appropriate port monitors.
- It 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.

Licensed Material – Property of copyright holder(s)

Once the port monitors are running, SAC polls them periodically for status information. The **sac**(1M) command line option, **-t**, allows the system administrator 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 will restart 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," below, and on the **sacadm**(1M) manual page in the *System Manager's Reference for the DG/UX™ System.*)

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. The system administrator 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* and in the **doconfig**(3N)manual page in the *System Manager's Reference for the DG/UX™ System* and placing this command script in the **_sysconfig** file. The per–system configuration script is interpreted by the Service Access Controller 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 the system administrator 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 the Service Access Controller 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 files allow the system administrator 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, the system administrator 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 version number of the Service Access Controller and an entry for the **tcp** port monitor. The system administrator adds 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.

NOTE: 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. The system administrator is 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, the system administrator has 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. 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.

NOTE: 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

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.

Figure 9–2 shows lines from a sample **ttymon** administrative file. 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," below.

Figure 9–2 shows the contents of a sample **ttymon** administrative file as listed by the **pmadm** command. The # character is a comment delimiter.

```
# 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:  -  #
```

Figure 9–2 Sample Contents of a ttymon Administrative File

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 Service Access Facility 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," below. 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 files are described under "Printing, Installing, and Replacing 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, that is, for port monitor administration (see the manual page **sacadm**(1M) in the *System Manager's Reference for the DG/UX™ System*). Under the Service Access Facility, port monitors are administered by using the **sacadm** command to make changes in SAC's administrative file. **sacadm** performs the functions listed below. Each function is discussed in one of the following sections.

- 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.

Printing Port Monitor Status Information

Unless the system administrator already knows the type of a port monitor, it may be necessary 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 the system administrator 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 **-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 command line in the following figure adds a port monitor of type **ttymon.**

```
# sacadm –a –p ttymonl –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" below. Per–system and per–port monitor configuration scripts allow the system administrator 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 the system administrator 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 hypothetical **_config** file in Figure 9–3, 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
```

Figure 9–3 Sample per–port monitor configuration script

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. If the files are edited by the system administrator, SAC and the port monitors are not notified. In this case, **sacadm** must be called with the **–x** option to notify SAC or port monitor of the changes.

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.

System administrators are advised against editing these files directly.

Appendix C contains a reference table that you may find useful for managing port monitors.

Service Management

The top level of the Service Access Facility is concerned with port monitor administration and is discussed in "Port Monitor Management" above. 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. Figure 9–4 shows sample **–l** output. In the figure, lines have been broken for readability.

```
PMTAG PMTYPE SVCTAG FLGS ID     PMSPECIFIC
tmon3 ttymon 6      ux   root   /dev/tty6 - - /usr/bin/login - 9600 - login: - \
                                #tty6
tmon3 ttymon 7      ux   root   /dev/tty7 - - /usr/bin/login - 9600 - login: - \
                                #tty7
tmon3 ttymon 8      ux   root   /dev/tty8 - - /usr/bin/login - 9600 - login: - \
                                #tty8
tmon3 ttymon 9      ux   root   /dev/tty9 - - /usr/bin/login - 9600 - login: - \
                                #tty9
tmon1 ttymon 1      ux   root   /dev/tty1 - - /usr/bin/login - 9600 - login: - \
                                #tty1
tmon1 ttymon 2      ux   root   /dev/tty2 - - /usr/bin/login - 9600 - login: - \
                                #tty2
tmon1 ttymon 3      ux   root   /dev/tty3 - - /usr/bin/login - 9600 - login: - \
                                #tty3
tmon1 ttymon 4      ux   root   /dev/tty4 - - /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 102    -    listen - c - /usr/tcp/lib/ttysrv -d -n ntty,tirdwr,ld0 \
                                #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
```

Figure 9–4 Output of pmadm –l

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 that the administrator selects.
–s *svctag*	Specify the tag for the port service. This is a name that the administrator selects.
–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 the system administrator 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 files. Only the per–service configuration files are interpreted by the port monitors. Per–service configuration scripts may be printed by any user on the system. Only the system administrator may 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.

The sample per–service configuration in Figure 9–5 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**.

```
runwait ulimit 4096
runwait umask 077
```

Figure 9–5 Sample per–service configuration script

Appendix C contains a reference table that you may find useful for managing port services.

The Port Monitor ttymon

ttymon is a port monitor invoked by the Service Access Controller (SAC). The Service Access Controller is the Service Access Facility's controlling process. It is started by **init** when the system enters multiuser mode. One of SAC's functions after it is started is to start all port monitors the system administrator has configured.

ttymon performs the functions that **getty** and **uugetty** performed in previous releases. Like **getty** and **uugetty**, **ttymon** sets terminal modes and line speeds for the port the user is connected to, allowing communication with the service associated with that port.

ttymon differs from **getty** and **uugetty** in several important ways:

- **ttymon** provides any service the system administrator configures. **getty** and **uugetty** provided only **login** service.
- Each invocation of **ttymon** can support multiple ports. **getty** and **uugetty** supported only one port per invocation.
- **ttymon** is a persistent process that continues to run after the service process is initiated. The **getty** and **uugetty** processes were replaced by the process of the service invoked.
- **ttymon** can take advantage of all STREAMS I/O capabilities.
- Line disciplines are configurable on a per–port basis.
- **ttymon** provides an optional autobaud facility that automatically determines the line speed of the hardware connected to any port monitored by a **ttymon** port monitor.

What ttymon Does

ttymon has three main functions:

- It initializes and monitors TTY ports.
- It sets terminal modes and line speeds for each port it monitors.
- It 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.

Licensed Material – Property of copyright holder(s)

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. This service can be any service configured by the system administrator. A typical example 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," below). 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 the Service Access Facility

The Service Access Facility (SAF) provides a generic interface to which all port monitors must conform. **ttymon** is a port monitor under the Service Access Facility's controller, the Service Access Controller. (See "Overview of the Service Access Facility," "Port Monitor Management," and "Service Management" for a description of the Service Access Facility, the administrative files it maintains, and the commands used for port monitor and service administration.) Figure 9–1 shows how a service, which may be a **login** service, is invoked using **ttymon**.

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 the Service Access Controller 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 in the **/etc/inittab** file. These entries contain calls to **ttymon** in "express." See "**ttymon** Express," below.

The ttyadm Command

The Service Access Facility 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 below 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

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" above, 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 a sample 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 above table, 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 'pmspecific-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," above, 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, the system administrator 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**.

NOTE: 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," above, 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 bidirectional.

```
# 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** bidirectional 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 ↵
```

will re–enable 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 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 the Service Access Controller 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.

Configuration Files

As a port monitor under the Service Access Facility, **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. Configuration scripts are installed by the system administrator, 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. The system administrator is 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: The Service Access Controller records aberrant port monitor behavior in **/var/saf/_log**; and each **ttymon** port monitor 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 ↵
```

will list the most recent 25 entries in the **_log** file.

Periodically, you should remove or truncate the log files. You can set up a **cron** job to clean up regularly. See "Automating Job Execution" in Chapter 2 and 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.

From the system administrator's point of view, the key elements in managing terminal line settings are the **ttydefs** file and the **sttydefs** command, which maintains the **ttydefs** file.

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. The figure following shows the relationship between the *ttylabel* and *nextlabel* fields in the **ttymon** administrative files and **ttydefs** files. The figure 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, the system administrator will be able to specify the default erase and kill characters.

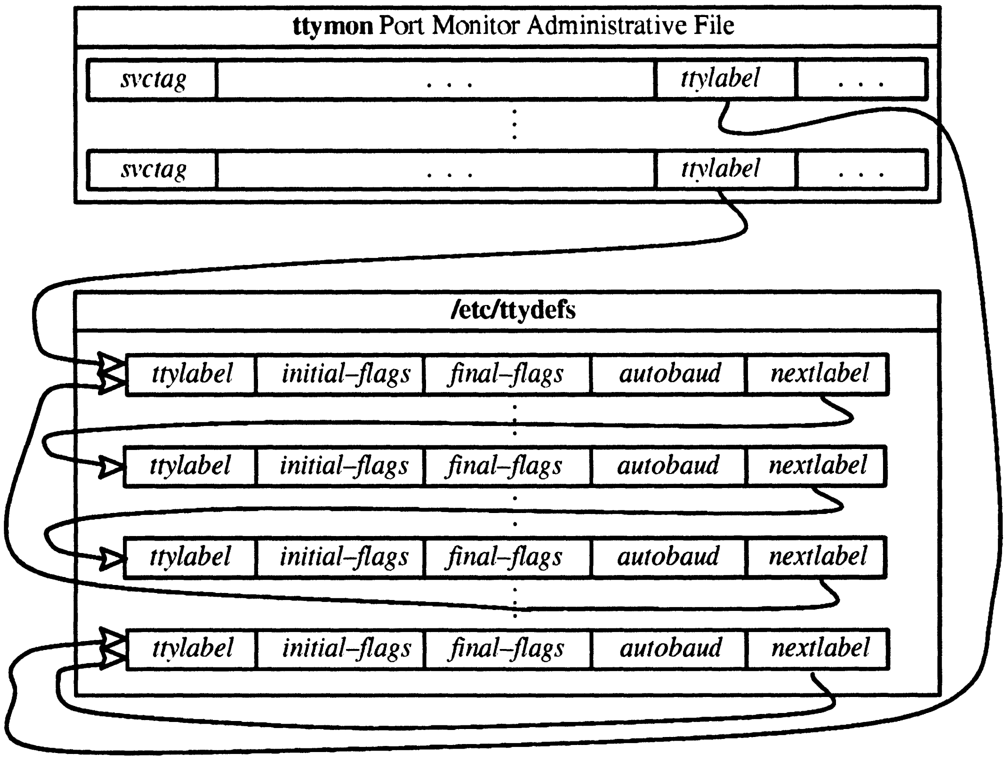

Figure 9–6 Port monitor/ttydefs links

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
```

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 describe 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.

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 pound sign (#). Most terminals transmit a **^h** when the **backspace** key is pressed.

Appendix C contains a reference table that you may find useful for setting terminal lines.

The Port Monitor listen

listen is a port monitor invoked by the Service Access Controller (SAC). The Service Access Controller is the Service Access Facility's controlling process. It is started by **init** when the system enters multiuser mode. One of SAC's functions after it is started is to start all port monitors the system administrator has 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:

- It initializes and monitors **listen** ports, and
- it invokes the service associated with a port in response to requests.

The **listen** port monitor also has these features:

- It allows private addresses for services,
- passes connections (file descriptors) to standing servers,
- supports socket–based services, and
- 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 the Service Access Facility

The Service Access Facility (SAF) provides a generic interface to which all port monitors must conform. **listen** is a port monitor under the Service Access Facility's controller, the Service Access Controller. (See "Overview of the Service Access Facility," "Port Monitor Management," and "Service Management," above, for a description of the Service Access Facility, 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 the Service Access Controller 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

The Service Access Facility 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 the 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

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,ld0 \
                                            #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," above, 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, the system administrator 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. **Do not edit 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 ↵
```

will re–enable port monitor **tcp**.

Configuration Files

As a port monitor under the Service Access Facility, **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. Configuration scripts are installed by the system administrator, 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.

Appendix C contains a reference table that you may find useful for managing **listen** port monitors.

End of Chapter

Chapter 10
Controller Management

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.

Sync Management

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, and Check 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. Except for the List operation, these operations are not appropriate for integrated synchronous controllers such as **iscd** and **izscd**.

Starting Sync Controllers

Use the 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 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 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 operation Device –> Sync –> List to display the **/dev** entries for sync controllers configured on your system, whether VME controllers or integrated controllers. The display may include entries such as **ssid**, **vsxb**, **iscd**, and **izscd**.

LAN Management

This section describes how to manage local–area network (LAN) controllers that execute controller–resident software downloaded from the host. The **sysadm** Device –> LAN menu contains operations for starting, stopping, and listing your LAN controllers.

You can use the LAN menu 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 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. The operation is available only if your system has configured intelligent controllers, which execute software downloaded from the host.

VDA Management

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.

End of Chapter

Chapter 11
Line Printer Management

This chapter tells how to set up and manage printers and print queues using the LP print service. First, the chapter covers the operations that you perform using the **sysadm** utility's Printer menu, which appears in the Device menu.

Second, the chapter covers expert material, the shell-level commands and files that you may use to manage the LP service. The **sysadm** Printer menu operations are implemented using the commands and files discussed in the expert section. The expert section includes troubleshooting tips.

For information on using the **lp** command, see the manual page for **lp**(1).

Printing Now

If you simply want to get a local line printer, local PostScript laser printer (serial or parallel), or a TermServer PostScript printer up and running now, follow these steps:

1. Make sure the printer is properly connected to your computer and powered on. See your hardware documentation.
2. Make sure your DG/UX kernel knows that you have a printer device on your system. If your printer uses a standard interface, such as **lp()** or **syac()**, that is included in your kernel, you may assume that the printer is accessible. If your printer is not a standard device or is connected to a controller that your kernel may not recognize, you need to rebuild the kernel. See Chapter 4 for information on building a kernel.
3. Find out which device pathname (such as **/dev/tty04**) is associated with your printer. You may have to refer to your hardware documentation or your installation manual for a discussion of ports, terminal lines, and device names. If you have a TermServer PostScript printer, the device is **/dev/null**.
4. Invoke **sysadm** and select the operation Device –> Printer –> Devices –> Add. Accept the default values for all queries except for these:

   `Name` Enter a name for the printer. The name may be any arbitrary name that is easy to type and remember.

   `Local Printer`
   Enter **yes**, even if a TermServer printer.

   `Device Path Name`
   Enter the pathname that you found in Step 3, above.

   `Printer Type`
   If you are adding a line printer, accept the default. If you are adding a PostScript printer, enter one of the following:

   `PS` Serial PostScript printer, normal paper stacking.

   `PS-r` Serial PostScript printer, reverse order paper stacking.

`PS-b` Parallel or TermServer PostScript printer, normal paper stacking.

`PS-br` Parallel or TermServer PostScript printer, reverse order paper stacking.

See "Adding Devices" for more printer types.

`Input Type`
If you are adding a line printer, accept the default. If you are adding a PostScript printer, enter `PS`.

`Interface`
For a TermServer PostScript printer, select `termprinter`. For other printers, select the default, `standard`.

Execute the operation. For a complete discussion of the queries in the Add operation, see "Adding Devices" in the next section.

5. Execute the operation Device -> Printer -> Devices -> Default, and make the new printer the default printer.

NOTE: Installers of TermServer PostScript printers need to add the printer's Internet address with the **sysadm** operation Networking -> TCP/IP -> Hosts -> Add. If possible, set your PostScript printer to operate in batch mode; otherwise, use the following command line to set the port on the TermServer to host mode:

```
# setd (!port_id) dv=(host, class, default) ↵
```

Now you should be able to print using the **lp**(1) command:

```
# lp  myfile ↵
```

LP Management Procedures

This section discusses the operations that **sysadm** offers for managing the LP service. For information on using **sysadm**, invoke **sysadm** and see the Help menu.

To set your LP service up for the first time, start by adding devices. See "Managing Devices." In the **sysadm** Printer menu, select the Devices menu. Before adding any remote printers, you need to add entries for remote hosts using the Systems menu.

The Printer menu comprises the following selections:

`Devices` This menu contains operations to add, delete, modify, and list entries for the printers on your system. These entries determine not only what your printers are called and how the LP service should communicate with them, they also determine other useful features such as what kinds of jobs the printers can accept, how the LP service should handle a printer fault, and so on. You can also name a printer on your system to be the default printer. You can set the LP service to accept or reject requests submitted to a given printer or class, and you can enable and disable individual printers.

`Classes` This menu contains operations to create, modify, remove, and list status of printer classes.

`Filters`	This menu contains operations to add, modify, delete, restore, and list filter programs for use with printers on your system.
`Forms`	This menu contains operations to create, modify, remove, and list definitions for pre-printed forms used on your printers. You can also mount forms on printers, which means configure a printer to print that form, and unmount forms from printers. In addition, you can display the current mount status of defined forms.
`Priorities`	This menu contains operations to set the default priority for print requests. You can also set and remove priorities for individual users, and you can set a default priority for print requests. A list operation displays current priority settings.
`Requests`	This menu contains operations to list and cancel job requests as well as to hold requests and to resume held requests. There is also an operation for moving requests from one printer to another.
`Scheduler/Service`	This menu contains operations to start and stop the LP service scheduler and to display the current status of the scheduler.
`Systems`	For systems on a network, this menu contains operations to add, modify, remove, and list entries for remote systems with which you can share printer services.
`List`	This operation displays the status of the LP service, including LP service database information about the default printer setting, devices, classes, remote systems, forms, filters, and current status of the scheduler, devices, classes, forms, and jobs requests.

You need to be superuser to use these operations except for the Requests menu operations that cancel, hold, and resume print jobs. Except for the List operation in the Filters menu, you do not need to be superuser to invoke any of the List operations.

The following sections discuss the Printer menu operations and LP services concepts in more detail.

Managing Devices

Before you can add printer classes to your system or perform other printer-related operations, you need to define the printers that are available on the system. These printers may be connected to the local system, or they may be on other systems in your network.

This section discusses the various operations that **sysadm** offers for printer device management:

`Add`	This operation adds a device entry to the LP service databases.
`Default`	This operation allows you to define a printer for requests that do not specify a destination.
`Delete`	Use this operation to remove the file, directories, and database entries that support a particular printer.
`Modify`	Use this operation to change the attributes of an existing printer.

`Accept`	This operation allows a printer to accept print requests.
`Reject`	This operation makes the LP service refuse requests for the printer. If a user submits a job for printing at this printer, the **lp** command returns an error.
`Disable`	This operation causes the LP service to cease sending jobs to a printer.
`Enable`	This operation tells the LP service to verify that a printing device is functional and if so to make it available for printing.
`List`	This operation produces a general status listing for your system's printing devices.

The following sections discuss these operations in more detail.

Adding Devices

If you are adding a remote printer or printer class, you need to verify that you have already added an entry for the remote system using the following operation Device –> Printer –> Systems –> Add.

After you have added the system entry, you may use the Add operation documented here to set up access to the printer.

To declare a printer device to the LP service, use the Add operation in the Devices menu. This operation not only sets up the files and directories that the LP service needs in order to use the printer, it also lets you customize the printer's behavior in terms of the various features that the LP service offers.

The Add operation contains a lot of prompts, but you do not need to respond to all of them if you simply want to get a printer up and running immediately. To make a printer available for immediate use, see "Printing Now" at the beginning of the chapter.

The Add operation includes queries covering a variety of LP service options. The following sections discuss these queries in more detail.

Printer name

A printer name may be any name that you choose. It is wise to choose a name that tells something about the printer, such as what type of printer it is, its location at your workplace, or its location on your network. It is also a good idea to choose a name that is short and easy to type.

Enable

Enabling the printer allows the LP service to send waiting jobs to it. If a printer is set to accept but it is disabled, the LP service accepts jobs submitted for the printer, but the LP service will not send the jobs to the printer for printing.

The Device menu provides operations for enabling and disabling printers.

Accept

Setting the printer to accept requests allows users to submit jobs for that printer. If a user submits a job for a printer that is set to reject jobs, the **lp** command does not queue the job; instead, it returns a reject message to the user.

The Device menu provides operations for setting printers to accept and reject job requests.

Local printer

If the printer is connected to your own computer system, it is considered local. If the printer is connected to another system on your network, however, it is considered remote. An exception to this rule, however, is a TermServer printer. Even though a TermServer printer is on the network, you should add it as a local printer. You should add a TermServer printer's Internet address on your system with the **sysadm** command Networking –> TCP/IP –> Hosts –> Add.

In addition to adding printers from remote systems, you can add classes from remote machines. To do so, follow the same procedures as for a remote printer. Once added, you access the remote class the same way you would access a remote printer. Any reference in this section to remote printers implies remote classes as well.

If you are adding a remote printer, you need to know several things about the remote system:

- You need to know the name of the host where the printer is connected. Network database files **/etc/ethers** and **/etc/hosts** should already contain entries for this host. If these files do not contain the needed entries, see the **sysadm** Networking menu. If you are in an NIS (Network Information Services) domain, see your NIS administrator. You should have already added an entry for the remote host using the Add operation of the Systems menu.
- You need to know the name of the printer, as defined on the remote system. This name does not need to be the same as the name that you give the printer locally.
- You need to know what version of scheduler the remote system uses. The scheduler may be either a DG/UX Release 5.4 LP scheduler, or it may be an earlier version.

If you are adding a remote printer, a number of the questions in this section do not apply to you at this time. Skip the sections for **Model**, **Device Path Name**, **Type**, **Stty Options**, **Print Options**, and **Input Types**. When you define a remote printer, you do not see these queries in the Add operation.

Device

You must supply a device pathname when adding a local printer. The pathname, indicating an entry in the **/dev** directory, must already exist.

The device pathname represents the port to which you have connected the printer. If your computer hardware has a dedicated line printer port, this pathname is **/dev/lp**.

If you have connected your printer to an asynchronous terminal line, you should refer to your installation documentation and/or your controller documentation to ascertain which line it is. An asynchronous terminal pathname has the form **/dev/tty***nn*, where *nn* is the line number.

If you have connected your printer to an asynchronous line, do not assign a port monitor service to that line. Use the Ports menu operations, make sure any service for the line is disabled.

If you are adding a TermServer printer, the device is **/dev/null**.

Printer type

A printer's type refers to a **terminfo** database entry that describes what control codes the printer requires for handling various initialization and configuration operations as well as printing attributes.

Initially, the default printer type is **printer–80**. Although you may leave the printer's type set to the default, you can enhance your printer's ability to serve users by assigning it a more appropriate type. For a complete list of supported types, review the list of **terminfo** database entries with this shell command line:

```
# ls –CR /usr/share/lib/terminfo/* | more ⤶
```

The following list contains some accepted printer types.

`printer`	Generic line printer; 132 columns.
`printer-80`	Generic line printer; 80 columns.
`lpr`	Same as printer type above.
`lpr-80`	Same as printer–80 type above.
`PS`	Serial PostScript printer, normal paper stacking.
`PS-r`	Serial PostScript printer, reverse order paper stacking.
`PS-b`	Parallel or TermServer PostScript printer, normal paper stacking.
`PS-br`	Parallel or TermServer PostScript printer, reverse order paper stacking.
`citoh`	Citoh 8510 printer.
`daisy`	Daisy brand printer.
`qume`	Qume Sprint 11.
`la100`	DEC LA 100 printer.
`ln03`	DEC LN03 laser printer.
`epsonfx`	Epson FX printer; 136 columns.
`epsonfx-80`	Epson FX printer; 80 columns.

If your printer can emulate multiple printers, you may specify more than one type. The **terminfo** entries for emulated printer types generally have names of the form *printer–emulation*, where *printer* is the model of printer and *emulation* is the type of printer that it emulates. For example, **dg6617–epsonfx** is a Data General model 6617 printer emulating an Epson printer.

Stty options

Specify additional **stty** options in this query. **Stty** options control line I/O characteristics such as baud, flow control, parity, and so on. For more information, see the **stty**(1) manual page.

If you are installing a TermServer PostScript printer, set your printer to batch mode if possible; otherwise, use the following command line to set the port on the TermServer to host mode:

```
# setd (!port_id) dv=(host, class, default) ⤶
```

Other print options

You may specify several print options to determine how the printer spaces jobs on the page. These options are:

`length`
: This option determines how many lines the printer will print per page. For example, to set page length to 63 lines, include the following in the option line:

```
length=63
```

`width`
: This option determines the width of the printable area on the page. Specify the width in characters. For example, to set page width to 40 characters, include the following option:

```
width=40
```

`cpi`
: This option determines the horizontal printing density, in characters-per-inch. For example, to set characters-per-inch to 10, specify the following option:

```
cpi=10
```

`lpi`
: This option determines the vertical printing density, in lines-per-inch. For example, to print 6 lines per inch, specify this option:

```
lpi=6
```

`banner/nobanner`
: The banner is the job header page. By default, the LP service prints a banner before every job. To allow users to request that a job be printed without a banner specify:

```
nobanner
```

When specifying these attributes for the Other Print Options query, list them separated by spaces, for example:

```
length=63 lpi=6
```

To set any of these printing attributes to their default values, specify them without a number, for example:

```
cpi=
```

In the case of the `banner` attribute, set it back to the default by specifying `banner`.

Input types allowed

You may set one or more input types, also called content types, for every printer you add. An input type is a designation telling what kind of files you can print on the printer. Initially, the default input type is **simple**, which means common ASCII character text.

Generally, most printers can print **simple** files and files whose input type is the same as the printer type. Additionally, you can make up other input types to correspond to printer types and the types of files you print. Here are some accepted input types:

`cif`	Output of BSD `cifpbt` processor.
`epsonfx`	Epson and compatible printers.
`fortran`	ASA carriage control format.
`otroff`	CAT typesetter instructions generated by BSD (old) `troff`.
`plot`	Plotting instructions for Tektronix displays and devices.
`PS`	PostScript language.
`raster`	Raster bitmap format for Varian raster devices.
`simple`	ASCII file.
`tex`	DVI format files.
`troff`	Device-independent output from the `troff` text-formatting processor.

Once you have assigned input types to your printers, users may submit requests specifying the input type with the **lp** command's **–T** option. As long as the request does not specify a destination and there is no default printer, the LP service will print the request on any printer that accepts the specified type. To override the LP service default printer, specify **–d** any on the **lp** command line.

For example, the following command line overrides the system default printer and submits the PostScript file **myfile** to any printer that accepts input type `postscript`:

```
% lp –d any –T postscript myfile  ↵
```

If adding a remote printer, list the same input types that are listed for the printer on its native system.

Device is also a login terminal

In some cases, the device used as a printing output device may also be used for logging into the system.

Fault recovery

The LP service can handle a printer fault in one of three ways:

`continue`	When the printer encounters a fault, it reprints the current page, clears the fault condition, and continues printing.
`beginning`	When the printer encounters a fault, it clears the fault condition and reprints the entire job.
`wait`	When the printer encounters a fault, the LP service stops the job and places the printer in a disabled state. The printer will not continue until you enable it. Enable a printer with the Enable operation in the Devices menu.

Alert messages

When a printer fault occurs, the LP service can alert you to the condition in several different ways:

`mail`	Using the **mail**(1) command, the LP service sends mail to the administrator.
`write`	Using the **write**(1) command, the LP service writes a message to the terminal where the administrator is currently logged in.

`quiet` Send no alert for the current print fault. You can select this option only if the LP service has already determined that your printer is in a fault state. This option is useful for silencing alerts for a fault condition that you already know exists.

`none` The LP service sends no alerts if it detects a fault.

As an alternative, you may write your own alert script. The script should accept the alert message as standard input from the LP service. When prompted, specify the command line for invoking your custom alert script.

You may also specify how often the LP service alerts you when a fault occurs. You specify the interval in minutes. For example, an interval of 10 means that the LP service will send out an alert every 10 minutes for as long as the fault condition exists. If you want the LP service to send out only one alert message per fault, specify 0.

Alert interval

When a printer fault occurs, the system sends periodic messages to alert the system administrator. The alert interval determines the number of minutes that the LP system will wait between messages. Specifying **0** indicates that you want only one message per fault sent to the administrator.

Users to allow

Optionally, you may specify which users may use the printer. Regardless of any users expressly denied access to the printer (via `Users to deny`, discussed below), only the users specified in `Users to allow` may use the printer. Specifying **all** allows access for all users except any who are denied access in `Users to deny`.

Users to deny

Optionally, you may specify which users may not use the printer. If you specify users in `Users to allow`, discussed above, it is not necessary to specify users in `Users to deny`.

Forms to allow

Specify forms, added with the Create operation of the Forms menu, that can be used on this printer. Regardless of any forms expressly restricted from the printer (via `Forms to deny`, discussed below), only the forms specified in `Forms to allow` may be used on the printer. Specifying **all** allows access use of all forms except any that are restricted in `Forms to deny`.

Forms to deny

Specify forms that may not be used on this printer. If you specify forms in `Forms to allow`, discussed above, it is not necessary to specify forms in `Forms to deny`.

Copy an existing printer's interface program

Select this option if you want this printer to use the same interface script or program that another local printer uses.

Model

The model of the printer refers to the interface script that the LP service uses to drive the printer. For a common line printer or laser printer not falling into one of the categories below, select the `standard` model. Other models are:

`dg455x`	For Data General text-only laser printers, models 4557 and 4558.
`remshlp`	For remote printers using a System V LP scheduler (the DG/UX default scheduler).
`remshlp_bsd`	For remote printers using a BSD spooler.
`remshlp_a`	For remote printers on an AOS/VS system.
`termprinter`	For TermServer line printers.

You may copy these interface scripts, located in **/var/spool/lp/model**, and tailor them to fit printers on your system. For more information about interface scripts, see "Expert Information" later in this chapter.

Printer Description

This is a comment field for recording any information you wish to record about the printer. The field may contain no more than 40 characters.

Setting the Default Printer

You can make one of your printers the default printer so that any job submitted without the destination (**–d** *destination*) option goes automatically to the default printer. The default printer may be either a local printer or a remote one.

Deleting Devices

The Delete operation removes the files, directories, and LP service database entries supporting the specified printer. Deleting an entry for a remote printer has no effect on the files, directories, and LP service database entries supporting the printer on the remote system. Deleting the last printer of a class also removes the class.

Modifying Devices

Use the Modify operation to change the attributes that you set for a printer when you added it. For a complete discussion of the various attributes, see "Adding Devices."

The Modify operation does not allow you to change the class to which a device belongs. To change the class membership, use the Modify operation of the Classes menu.

Accepting or Rejecting Requests

Use the Accept operation to allow users to submit requests to a printer device. To make the LP service reject requests for a given printer, use the Reject operation.

When a user submits a request to a printer that is rejecting requests, the LP service returns an error.

Setting a printer to reject requests does not remove jobs for that printer that are already in the print queue.

The accept or reject status of a printer has no effect on its enable or disable status, described below.

Enabling and Disabling Devices

The Enable operation makes a printer available to print requests. If the fault recovery attribute for a printer is set to "wait" (see "Adding Devices"), you will have to enable a printer after a fault occurs.

The Disable operation makes a printer unavailable to print jobs. Users may still submit jobs for printing by the printer even though it is disabled (assuming it is set to accept requests). Requests submitted for a disabled printer will remain in the print queue until you remove them, move them, or enable the printer.

The Disable operation allows you to select how to handle a job currently being printed by the printer. You may choose to interrupt the job, in which case the operation cancels the job, removing it from the queue, or you may choose to let the printer finish the request before entering the disabled state.

When you disable a printer, you may specify a reason in the form of a line of text of your own choosing, that tells why you have disabled the printer. When a user displays the status of the printer, the display includes the reason that you entered.

Displaying Devices

Use the List operation to display information about any or all printers that you have added on your system. When you invoke the List operation, you may select either of two kinds of listing:

`setup` This listing shows not only some information about the current state of the printer, but it also displays the settings of attributes that you defined for the printer when you created it with the Add operation:

Forms mounted
Content types
Printer types
Description
Connection
Interface
On fault
After fault
Users allowed
Forms allowed
Banner
Character sets

Default pitch
Default page size
Default port settings

See "Adding Devices" for a discussion of these attributes.

`state` This listing shows only information pertinent to the current state of the printer, whether it is enabled and accepting requests, and how many requests are in its queue. The display also includes the device pathname.

Managing Classes

A printer class is a group of printers to which you may submit a print request for printing by any printer in the class. When you specify a class on the **lp** command line, the LP service prints the job on the first available printer in the class. The advantage to this system is that users do not need to investigate the print queue every time they submit a job in order to see which printer is currently available or which printer already has jobs queued up. The result is faster throughput for your users and more efficient utilization of your printers.

Another advantage of print classes is that they allow you to maximize use of favored printers. For example, if you have a fast line printer and a slow line printer, you can create a class where the fast line printer is first and the slow one is second. Thus, the LP service submits the job to the fast printer if both are available and to the slow printer only when the fast printer is already in use.

Creating Classes

The Create operation of the Classes menu creates a new printer class and allows you to add printers to the class. When you create the class, you must add at least one printer to it. The printers you add should already exist; add printers using the operation Device –> Printer –> Devices –> Add.

A class may contain only local printers. See "Managing Devices" in this chapter for more information.

When you create a class, the order in which you list the printers determines the order in which the LP service checks them when assigning jobs. When you submit a job to the class, the LP service assigns the job to the first printer in the list that is available. A printer may belong to more then one class.

The Create operation also allows you to specify whether you want the class to accept jobs or reject jobs. For normal operation, specify Accept. If you do not want users to use the class at this time, specify Reject. Even if you set a class to reject requests, a user can still use a printer in the class by specifying the printer name on the **lp** command line (using the **–d** option).

To submit a job for printing by a certain class, use the **–d** option of the **lp** command. For example, to print the file **myfile** at the first available printer in class **class1**, use this command:

```
% lp –d class1 myfile ↵
```

Modifying Classes

Use the Modify operation of the Classes menu to modify a printer class. The class must already exist.

The Modify operation allows you to add printers to the class or remove members from the class. If you add a member to the class, the printer must already exist. Add a printer with the operation Device –> Printer –> Devices –> Add.

To change the order of members in the class, you need to remove them and add them again in the desired order. The order of printers in the class determines the order in which the LP service checks them when assigning jobs. When you submit a job to the class, the LP service assigns the job to the first printer in the list that is available.

While you are adding or removing printers, you can also change the accept or reject status of the class. Even if you set a class to reject requests, a user can still use a printer in the class by specifying the printer name on the **lp** command line (using the **–d** option).

Removing Classes

Use the Remove operation to delete a printer class from the LP service. You do not have to delete the printers from the class before removing the class.

Displaying Classes

To display information about the classes on your system, select the List operation from the Classes menu. You may list the status of all classes on the system, or you may specify a class for listing. The operation lists the printers that are members of the classes.

Managing Filters

Filters are programs that process files before printing them. This section covers the **sysadm** operations, located in the Filters menu of the Printer menu, that you use to manage LP service filters.

A filter can function in three ways:

- To convert a file from one format to another before printing.
- To handle the kind of special printing modes that a user can request with the **lp** command's **–y** option: two-sided printing, landscape printing, draft quality printing, and so on.
- To detect printer faults and notify the LP service.

When defining a filter, you not only name the program that functions as the filter, you also specify what content types the filter will accept as input and produce as output.

A content type refers to the formatting, codes, or conventions used in a file to describe its page layout or contents. The purpose of having content types is to allow the LP service a means of matching user-submitted print requests with compatible filters and printers. When a user submits a job and specifies a content type, the LP service attempts to match the file's content type with the printer type or input type specified for a printer. If no satisfactory printer exists, the LP service looks for filters that can convert the file into a printable content type.

For example, you may have created filters that accept or produce PostScript content. If you submit a file for printing and you specify that the file is of type PostScript, the LP service will look for a printer whose input type is PostScript. If the print service does not find one, it will attempt to assemble a series of filters that can take your PostScript file and convert it to a type that one of your printers can print.

You set a printer's input content type when you add it with the Add operation in the Devices menu. To change a printer's input types, use the operation Device -> Printer -> Devices -> Modify.

The Filters menu provides operations for:

- Adding filters.
- Modifying filters.
- Removing filters.
- Restoring filters.
- Displaying filters.

The following sections elaborate on these operations in more detail.

For information on writing a filter, see the section “Providing Filters” in the “Expert Information” section of this chapter.

Adding Filters

Use the Add operation to introduce a filter program to the LP service. When invoking the Add operation, be prepared to supply the following information:

- The command line for invoking the filter.
- The existing filter, if any, to copy to make the new filter.
- The content types the filter will accept as input.
- The content types the filter will produce as output.
- The printer types compatible with the output types.
- The printers that may print the filter's output.
- The filter speed.
- The options to use when invoking the filter.

The following sections elaborate.

Command Line

This is the full pathname of the filter program. If there are any fixed options that the program always needs, include them here.

Filter to Copy

This is any existing filter that you may want to copy to make the new filter.

Input Types

This is the list of file content types that the filter can process. The content type designations are names that you make up based on the types of files your users print and the types of printers you have. The LP service matches file content types and filter output types with filter input types, so be careful to use consistent naming conventions. Most filters can accept only one content type as input.

Output Types

This is the list of file content types that the filter can produce. The LP services matches content output types with filter input types and printer input types, so, again, use consistent naming conventions. Most filters can produce only one content type as output.

Printer Types

This is the list of printer types for which the filter may produce output. In most cases, this list will be the same as the Output Types list. Any listed printer types should match types of existing printer devices.

Printers

There may be printers who are compatible with the filter's output type but for other reasons are undesirable as output devices for this filter. When this case is true, you should specify the desirable printers in the Printers list, omitting the undesirable ones of the same type.

Filter Speed

You can designate a filter as either fast or slow. When a print request requires a filter designated "fast," the print service assigns a printer to the request at the same time it starts the filter. If a filter is particularly slow, however, this implementation occupies the printer unnecessarily because the printer is out of use while it waits for the filter to finish.

To avoid this kind of waste of printer time, you can designate such filters as "slow." The LP service executes slow filters in the background without causing the printer to wait; the slow filter does not access the printer (or the next filter in the series) until it has filtered the entire request. While the slow filter works, other requests are free to print.

If you are adding a filter that is intended to detect printer faults, you must designate the filter as "fast." You may designate a filter as "slow" if it does not require access to the printer.

Slow filters that are invoked by modes (via the **lp** command's **–y** option) must be run on the system where the print request was issued. The LP service cannot pass values for modes to remote systems. It can, however, match a file content type (specified after the **–T** option) to a content type on a remote system. Therefore, to activate special modes on a remote system, specify content types that will cause the LP service to match input types and output types.

Options

The Options field allows you to specify how user options, LP service database settings, and preceding filter options can determine options to be passed to this filter.

For example, you can set the options field so that if a specified character set is in effect for this print request, the LP service should include a particular option on the filter command line.

Specifically, the Options field allows you to determine the filter invocation command line based on these printing attributes:

- Input content type.
- Output content type.
- Printer type.
- Printer name.
- Character pitch (characters per inch).
- Line pitch (lines per inch).
- Page length.
- Page width.
- Pages to print.
- Character set.
- Form name.
- Number of copies.
- Special modes.

For a detailed discussion on how to set the Options field for these attributes, see "Defining Options with Templates" in "Expert Information."

Modifying Filters

Use the Modify operation to change the definition for an existing filter. For a discussion of the features that make up a filter, see "Adding Filters."

Removing Filters

Use the Remove operation to delete the LP service database entries that support a given filter. The operation does not remove the filter program itself.

Restoring an Original Filter Definition

Use the Restore operation to restore a filter's definition to the definition that originally shipped with the system. This operation functions only for filters that shipped with the DG/UX system.

Displaying Filters

This operation displays the attributes of filters currently available on your system. These attributes include:

`Filter command name`	The command name that invokes the filter.
`Input content type`	The list of content types that the filter accepts as input.
`Output content type`	The list of content types that the filter produces as output.
`Printer type`	The list of printer types for which the filter produces output.
`Printer names`	The list of printers for which the filter produces output.
`Speed`	The filter's speed.
`Command`	Any options that the LP service passes to the filter..

The "Adding Filters" section, earlier in this chapter, discusses these attributes in more detail.

Managing Forms

A form is a description of a page layout, plus some other attributes, that determine how a printer loaded with pre-printed form stock should complete special requests. For example, if you have a printer loaded with pre-printed purchase order forms, the form description provides the printer with descriptive information such as lines per inch, characters per line, required print wheels, ribbon colors, and so on. The form description can also include an alignment pattern that you can print to make sure that the form stock is aligned correctly in the printer.

Once you have defined a form, you associate it with a printer by mounting it on that printer. The LP service then restricts use of the printer to printing requests that require that form. If a user submits a job requiring a form that is not mounted on the desired printer, the system alerts you with a message. You can determine where the alert message goes (who receives it) when you define the form.

To submit a job for a particular form, include the **–f** option on the **lp** command line. For example, if you have a file called **PO–3992** that is formatted to be printed as a purchase order using a form you call **porder**, print the file on the form with a command line like this:

```
% lp  -f  porder  PO-3992  ⤶
```

The Forms menu offers several operations for managing forms:

`Create`	Create a new form definition.
`Modify`	Change an existing form definition.
`Remove`	Delete a form description from the LP services databases.
`Mount`	Assign a form restriction to a printer.
`Unmount`	Remove a form restriction from a printer.
`List`	Display information about the forms currently defined on the system.

`Show Status` Display the current status of forms on the system.

The following sections elaborate.

Creating Forms

Select the Create operation to enter a forms description. The operation asks a number of questions having to do with the desired page layout and other printing attributes. Some of these attributes, characters per line, for example, depend on the capabilities of your printer. You may need to consult your printer documentation to see what it can do and what it cannot do.

The following sections discuss the information you may specify when you define the form. The default form contains values for printing a normal page of 66x80 ASCII characters.

Name

The form name is any arbitrary name that you may choose. You should select a name that describes the form sufficiently but is still easy to type.

Form to Copy

When defining a form, you may specify an existing form whose definition you wish to start with as a model. If you need to define numerous forms that are similar to each other but that do not use many of the **sysadm**'s preset form defaults, you can create your own default form and copy the others from it.

Page Length

This value specifies the length of the form. For multi-page forms, this value is the length of each page. Specify the length as a number of lines, or in inches or centimeters. To specify a length in inches or centimeters, follow the value with an **i** or **c**, as appropriate. For example, specify six inches as **6i**.

Page Width

This value specifies the width of the form. Specify the length as a number of characters, or in inches or centimeters. To specify a length in inches or centimeters, follow the value with an **i** or **c**, as appropriate. For example, specify 20 centimeters as **20c**.

Number of Pages

This value specifies the number of pages making up a multi-page form.

Lines per Inch

Also called line pitch, this value determines how many lines appear per vertical inch. You may specify this value in either lines per inch or lines per centimeter by following the value with an **i** or a **c**, respectively. If you omit the letter, the operation assumes that you mean inches.

Characters per Inch

This value represents character pitch, the number of characters printed per horizontal inch. You may specify this value in either characters per inch or characters per centimeter by following the value with an **i** or a **c**, respectively. For example, specify 3 characters per centimeter with **3c**. If you omit the letter, the operation assumes that you mean inches.

Print Wheel

For this field, you may make up a name to represent a particular print wheel, character set, or font cartridge required to print the form. Make the name descriptive but easy to type. You should be consistent in your naming conventions.

Users can submit print requests specifying a particular print wheel or character set by including the **–S** option on the **lp** command line. If the required print wheel or character set is not mounted on the destination printer, the LP service will hold the job and alert you that you need to mount the print wheel or character set.

Ribbon Color

If the form requires a particular color of ribbon, you specify it in this field. Whenever you mount the form, the LP service will remind you to load the correct color of ribbon. The LP service does not track ribbons; therefore, it will not alert you if the wrong ribbon is loaded on a printer. It is up to you make sure the correct ribbon is loaded.

Comment

The comment field is for you to enter any information you wish about the form. Users can display the comment for a form; therefore, it is useful for you to enter a comment describing the form, its purpose, and so on.

Alignment Pattern

The alignment pattern is any pattern of characters that you may print to see if the form stock is loaded correctly in the printer. For security reasons, the LP service only allows the superuser and itself to see the alignment pattern.

Alert Messages

When a user submits a request requiring a form that is not mounted on a printer, the LP service alerts you. The alert may occur several different ways:

`mail`	Using the **mail**(1) command, the LP service sends mail to the administrator.
`write`	Using the **write**(1) command, the LP service writes a message to the terminal where the administrator is currently logged in.
`quiet`	Send no alert for the current form need. This option is useful for silencing alerts for a form need that you already know exists. Selecting `quiet` does not change the current alert status if there is currently no form need.
`none`	The LP service sends no alerts if it detects a need for a form.

As an alternative, you may write your own alert script. The script should accept the alert message as standard input from the LP service. When prompted, specify the command line for invoking your custom alert script.

You may also specify how often the LP service alerts you when a form need arises. You specify the interval in minutes. For example, an interval of **10** means that the LP service will send out an alert every 10 minutes for as long as the need exists. If you want the LP service to send out only one alert message upon detecting a need for a form, specify **0**.

Job Threshold

If you do not want to receive an alert message every time someone needs a form mounted, you can set a threshold determining how many requests needing a form will occur before the LP service alerts you. For example, setting the threshold to **5** means that the LP service will not alert you to mount a form until it has received 5 print requests needing a form.

Users to allow

Optionally, you may specify which users may use the form Regardless of any users expressly denied access to the form (via `Users to deny`, discussed below), only the users specified in `Users to allow` may use the form Specifying **all** allows access for all users except any who are denied access in **Users to deny**.

Users to deny

Optionally, you may specify which users may not use the form. If you specify users in `Users to allow`, discussed above, it is not necessary to specify users in `Users to deny`.

Modifying Forms

Use the Modify operation to change the description for an existing form. For a discussion of the fields in the form description, see "Creating Forms."

Removing Forms

To remove the description of a form from your system, select the Remove operation.

Mounting and Unmounting Forms

The LP service tracks forms status by allowing you to mount and unmount forms whenever you load or unload, respectively, a form on a printer. After loading form stock on a printer device, select the Mount operation to tell the LP service and users that the form is now available. With the form mounted, the LP service will print requests requiring the form.

After unloading form stock from a printer, use the Unmount operation to tell the LP service and users that the form is no longer available. If a user submits a request that requires a form that is not mounted, the LP service holds the job in the queue without printing it, and it sends you an alert message. The nature of the alert message depends on the Alert Messages option that you selected when you added the form. For more information on these options, see "Creating Forms," earlier in the chapter. To change the way the LP service currently handles alert messages, select the Modify operation.

To see what forms are mounted on what printers on your system, select the Show Status operation.

Displaying Forms

The List operation displays the form description attributes that you set with the Create operation. These attributes are:

`Name`	The name that you gave the form. This line in the display also tells whether or not the form is available for you to use.
`Page Length`	The length of the page in lines, inches, or centimeters. Values in inches and centimeters are followed by **i** or **c**, respectively.
`Page Width`	The width of the page in characters, inches, or centimeters. Values in inches and centimeters are followed by **i** or **c**, respectively.
`Number of Pages`	The number of pages making up the form.
`Line Pitch`	The number of lines per vertical inch.
`Character Pitch`	The number of characters per horizontal inch.
`Character Set`	The character set, print wheel, or font cartridge required by the form.
`Ribbon Color`	The color of ribbon required for the form.
`Comment`	A comment that you supplied when you added the form. This comment may describe the form, its purpose, and so on.
`Alignment`	A pattern of characters that you can print to see if the form stock is loaded correctly in the printer. For security reasons, only the superuser and the LP service can access and display the alignment pattern.

For a more detailed discussion of these attributes, see "Creating Forms."

Showing Status of Forms

Use the Show Status operation to see what forms on your system are currently mounted and available to you.

Setting User and Request Priorities

The LP service offers several different ways of controlling which jobs print first when multiple jobs are competing for printing resources. This priority mechanism revolves around the concept of a priority level associated with each job. Very simply, the lower the priority level number, the higher the priority of the request. Conversely, requests with higher priority numbers have lower priority. The LP service prints jobs with lower priority level numbers first.

Using the **–q** option of the **lp** command, users can set priority numbers for the jobs they submit. Priorities range from 0 (high priority) to 39 (low priority). For example, to submit a request to print file **nicefile** at the lowest priority, use this command:

```
% lp –q 39 nicefile ⮐
```

To help regulate the use of the priority system, the LP service allows you to set limits for priorities that users can specify. The Priorities menu provides these operations:

`Job Default`	Set a default priority for requests that do not specify a priority with the **–q** option.
`Remove`	Remove a default job priority limit for a specific user.
`Set`	Set a default job priority limit for a specific user.
`User Default`	Set a default job priority limit for all users who do not have an individual limit.
`List`	List current job priority limits for general requests and for specific users.

The following sections describe these operations in more detail.

Job Default

Use the Job Default operation to set the priority for all requests submitted without the **–q** option. You use the **–q** option on the **lp** command line to set the priority for a job.

Remove

Use this operation to remove the priority limit set for an individual user. You set an individual limit with the Set operation. Once you have removed the individual limit, the user's requests are subject to the User Default, if set.

Set

This operation sets a priority limit for an individual user. The lower the priority level, the higher priority job they may submit. Regardless of a user's priority limit, if the user does not specify a priority with the **–q** option, the request takes the system default priority level.

User Default

Use this operation to set a priority limit for all users for whom you have not set an individual limit. Regardless of the user default priority limit, if a user does not specify a priority with the **–q** option, the request takes the system default priority level.

List

This operation displays the priority values set with the other Priorities menu operations.

Managing Requests

The Requests menu offers several operations for handling jobs in the print queue. Unlike other printer management operations (other than list operations), some of these operations are accessible to users other than the superuser. The following sections elaborate.

Canceling a Request

Use the Cancel operation to remove a request from a queue. The superuser can cancel any request, but other users can cancel only their own requests.

Licensed Material – Property of copyright holder(s)

Holding a Request

Any user can use the Hold operation to suspend his or her own request. A held request remains in the queue, but the LP service will not send it to a printer until the user (or superuser) releases the request with the Resume operation. A held request does not block the print queue: the LP service will continue to serve other requests.

Resuming a Held Request

Use the Resume operation to release a request that a user suspended with the Hold operation. Users other than the superuser can resume only their own requests.

Moving Requests

The superuser can use the Move operation to move requests from the queue for one printer or class to the queue for another printer or class.

Displaying Requests

Use the List operation to display the requests currently in the print queue. Like other List operations in **sysadm**, any user may use this operation.

Managing the Scheduler

The LP scheduler is the process that manages print queues and the other LP services. The system starts the LP scheduler when you bring the system up to run level 1. The Scheduler menu offers three operations, described below.

Starting the Scheduler

Use the Start operation to begin execution of the LP scheduler. This operation performs no function if the scheduler is already running.

Stopping the Scheduler

Use the Stop operation to halt execution of the LP scheduler. This operation performs no function if the scheduler is already stopped.

This operation does not remove jobs from the print queue. When you restart the scheduler, it resumes processing queued requests.

Displaying Scheduler Status

Use the List operation to display the status of the LP scheduler and the print queues. The operation tells whether or not the scheduler is running, and it displays the queue for each printer on the system. In a queue display, the entry for each request shows:

- The request number.
- The user who submitted the request.

- The size of the file in bytes.
- The time and date when the job was submitted.
- Any printer specified.

Managing Remote Systems

If your system is on a network, the LP service allows you to share printer services with other systems in your network. Use the operations in the Systems menu to manage the LP service databases that contain information about remote systems.

Before adding a remote printer that resides on a system using a DG/UX Release 5.4 LP scheduler, add an entry for the remote system using the Add operation in the Systems menu. On the remote system, the system administrator must do the same, adding similar information about your system. After adding this information on both systems, you can set up access to the remote system's printers, and the remote system administrator can set up access to your printers.

The following sections discuss the operations in the Systems menu.

Adding Remote Systems

Use this operation to add the required LP service database entries that allow you to share printing resources with a remote system. To add a remote system, you need to do several things:

`Name` See the administrator of the remote system to verify the system's host name. If your systems are not part of an NIS (Network Information Services) domain, you need to make sure that entries for the host exist in your system's **/etc/ethers** and **/etc/hosts** files. To add entries to these files, see the **sysadm** utility's Networking menu.

If you are in an NIS domain, the required entries may already exist in the NIS databases. If not, see your NIS administrator.

`Scheduler Type`
You need to know if the remote system's scheduler is compatible with the AT&T System V scheduler (as is the DG/UX Release 5.4 scheduler) or with the BSD scheduler. If you are not sure, try the AT&T scheduler (**s5**) first.

`Connection Timeout Period`
Set a time in minutes that the connection with the remote system may remain idle before "timing out," or terminating.

`Connection Retry Interval`
Set the number of minutes to wait, after unexpected disconnection of service from the remote system, before attempting to reconnect.

`Comment`
Decide on an optional comment for the system entry. The List operation will display this comment with the other system information.

Modifying Remote Systems

Use the Modify operation to change the attributes associated with a system entry. See "Adding Systems" for information about the entry attributes.

Removing Remote Systems

The Remove operation deletes the remote system entry from the LP services databases. This operation does not remove entries from the **/etc/ethers** or **/etc/hosts** files.

Displaying Remote Systems

Use the List operation of the Systems menu to display remote system entries added with the Add operation. The section on the Add operation, above, describes the fields in the entry.

Displaying LP Service Status

The List operation of the Printer menu displays the status of the entire LP services subsystem. The display includes:

- Whether the scheduler is running.
- Which printer, if any, is the default destination.
- The device/printer name assignments.
- The accept status of each printer and the time and date when the accept status last changed.
- Whether the printer is enabled or disabled and the time and date when the enabled/disabled status last changed. If disabled, the report includes the reason.
- Which forms, if any, are available to you, and where they are mounted.
- Requests currently in the queue.

Expert Information

This section describes the shell-level commands that **sysadm** uses to provide LP services. Table 11–1 shows which **sysadm** menus and shell commands are available for administering the LP service.

Table 11–1 LP Print Service Menu and Command Summary

Task Description	Menu Selection	Shell Command
Configure printers for print service, set the default printer, turn queuing of requests on and off, enable and disable printers, display state and configuration of printers	**Devices**	**lpadmin**(1M), **accept**(1M), **reject**(1M), **enable**(1), **disable**(1), **lpstat**(1)
Group printers into classes	**Classes**	**lpadmin**(1M)
Provide pre-processing software for files to be printed	**Filters**	**lpfilter**(1M)
Define pre-printed forms for print requests	**Forms**	**lpforms**(1M)

Continued

Table 11–1 LP Print Service Menu and Command Summary

Task Description	Menu Selection	Shell Command
Define levels of priority available to users requesting print jobs	**Priorities**	**lpusers**(1M)
Cancel, hold, resume, move, and list print requests	**Requests**	**lp**(1), **cancel**(1), **lpmove**(1M), **lpsched**(1M)
Start and stop print service, report status of scheduler and list print requests	**Scheduler/Service**	**lpsched**(1M), **lpshut**(1M), **lpstat**(1)
Set up communication to remote print service	**Systems**	**lpsystem**(1M)
Identify active printers, print wheels & character sets, mounted forms, and pending requests	**List**	**lpstat**(1)

The rest of this section describes the work required to set up and maintain print services with the LP print service utilities. Details about the commands listed above are available in the manual pages for them.

This section includes the following information:

- A description of how the LP print service works
- References to documentation for installing the print service
- Troubleshooting guidelines
- Instructions for stopping and starting the print service manually
- Instructions for configuring a print service for the unique requirements of your users (such as the need for particular pre-printed forms and filters)
- A list of directories and files delivered as part of the print service
- Instructions for supporting PostScript printers
- Instructions for writing customized filters and interface programs

Overview

The LP print service, originally called the LP spooler, is a set of software utilities that allows you, minimally, to send a file to be printed while you continue with other work. The term "spool" is an acronym for "simultaneous peripheral output on-line," and "LP" originally stood for Line Printer, but has come to include many other types of printing devices. The print service has many optional enhancements; however, you can make your print service as simple or as sophisticated as you like.

Components of a Print Service

A print service consists of both hardware and software. You must have at least one computer and one printing device for an LP print service. Beyond that, you may have any number of computers and printing devices; there is no limit to the number of pieces of hardware you may include. The software consists of the LP print service utilities and any filters (programs that process the data in a filebefore it is printed) that you may provide. Users of your print service may be required to print all their files in the same format, or, if you make different types of printers and/or filters available with your service, they may choose from several formats. You may also offer your users a choice between plain paper and pre-printed forms (such as invoices or checks).

Functions Performed by the Print Service Software

Whether your print service is simple (such as a one-computer/one-printer configuration that prints every file in the same format on the same type of paper) or a sophisticated one (such as a computer network with multiple printers and a choice of printing formats and forms), the LP software helps you maintain it by performing several important functions:

- Scheduling the print requests of multiple users
- Scheduling the work of multiple printers
- Starting programs that interface with the printers
- Filtering users' files (if necessary) so they will be printed properly
- Keeping track of the status of jobs
- Keeping track of forms and print wheels currently mounted and alerting you to mount needed forms and print wheels
- Alerting you to printer problems

Suggestions for LP Print Service Administration

Here are some tips about how to organize your duties as the administrator of an LP print service.

Configuring Your Printer Sites

Where you decide to put your printers and how you decide to connect them to your computers depends on how those printers will be used. There are three possible scenarios: (1) users may access printers attached to their own computer; (2) users may access printers attached to a server computer; and (3) users may access remote printers on a network to which their computer belongs.

- You may want to connect a particular printer directly to the computer that is the home system of the users who will use that printer most often. If you do, the type of connection you have will be referred to as a direct connection. An environment that includes more than one computer, each of which has a direct connection to a printer, is said to have a "distributed printing configuration."
- You may want to have all your printers in one physical location, such as a computer center. If so, you might connect them all to one computer. Users on other computers who want to use a

printer may access it through a network linking their own computers to the computer serving the printers. An environment in which one computer serves several printers (which can be accessed only through a computer-to-computer network) is described as a "print server configuration."

Figure 11–1 shows a sample print server configuration.

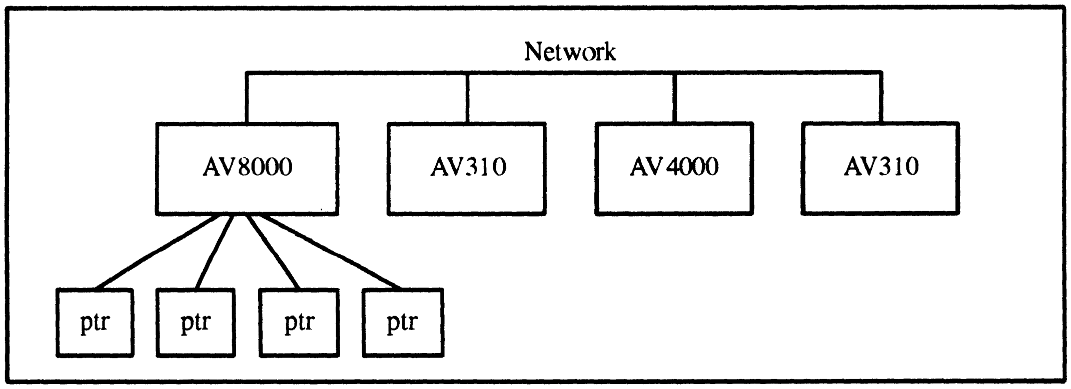

Figure 11–1 Print Server Configuration

You may want to link most of your printers to a dedicated printer server computer, while allowing other printers to be connected to your system. If so, you can arrange your computers and printers in a network configuration as shown in Figure 11–2.

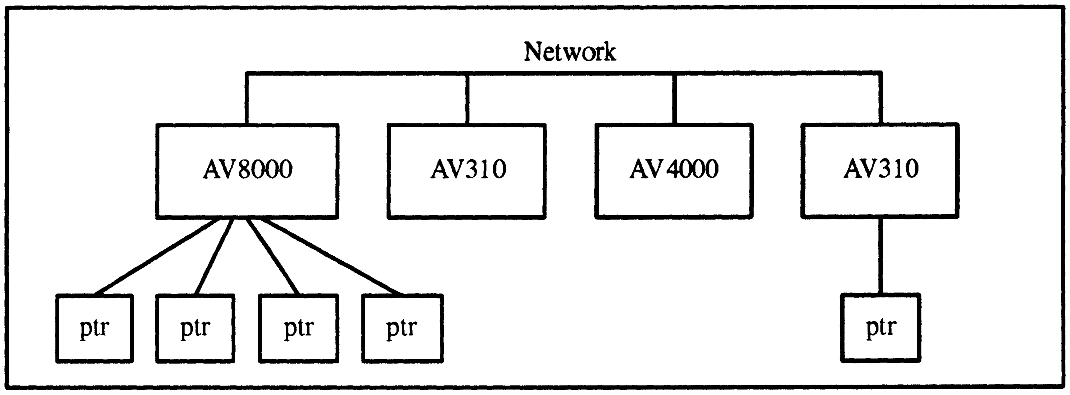

Figure 11–2 Network Configuration

Configuring Your Printers

Before the LP print service can start accepting print requests, you will have to describe the characteristics of each printer you have. The following is a list of the attributes most commonly defined:

- printer name (mandatory)
- connection method (mandatory for local printers)
- system name (mandatory for access to remote printers and mandatory for allowing remote access to local printers)

- interface program
- printer type
- content types
- printer port characteristics
- character sets or print wheels
- alerting to mount a print wheel
- forms allowed
- printer fault alerting
- printer fault recovery
- restrictions on user access
- inclusion of banner page in output
- printer description
- default printing attributes
- printer class membership
- system default destination
- mounting a form or print wheel
- removing a printer or class

You need to specify very little of this information to add a new printer to the LP print service. The more information you provide, however, the better the printer will satisfy various users' needs.

The descriptions in the sections below will help you understand what this printer configuration information means and how it is used, so that you can decide how to configure your printers. In each section you will also be shown how to specify this information when adding a printer. While you can follow each of the sections in order and correctly configure a printer in several steps, you may want to wait until you've read all the sections before adding a printer, so that you can do it in fewer steps.

Printer Name

The printer name and the connection method (described next) are the only items you must specify to define a new local printer. To define a new remote printer, you must specify the printer name and the system name. The printer name is used to identify the printer, both by you (when you want to change the printer configuration or manage the printer), and by users who want to use the printer. The name may contain a maximum of 24 alphanumeric characters and underscores.

You may choose any name you like, but it is good practice to choose a name that is meaningful to the users of the LP print service. For example, **laser** is a good name for a laser printer. If you have several laser printers you may name them **laser1**, **laser2**, and so on.

You don't have to try to fit a lot of descriptive information into the name; there is a better place for this information (see "Printer Description" below). You also don't have to make the name precisely identify the type of printer; users who need to use a particular type of printer can specify it by type rather than name (see "Printer Type," below).

You will use the printer name every time you want to refer to the printer: when adding other configuration information for the printer, when changing the configuration of the printer, when referring to the status of the printer, and when removing the printer. Thus the first thing you must do to add a printer is identify its name. You will do this as shown below; but don't do it yet because you'll also need to specify the connection method.

```
# lpadmin -p printer-name ↲
```

There are no default names; you must name every printer.

Connection Method

This section does not apply if you are making a remote printer (one that is connected to a remote system on your network) accessible to users on your system. The LP print service allows you to connect a printer to your computer in one of the following three ways:

- by connecting the printer directly to your computer
- by connecting the printer directly to a network to which your computer is attached
- by connecting the printer to a modem.

Figure 11–3 shows these three types of connections.

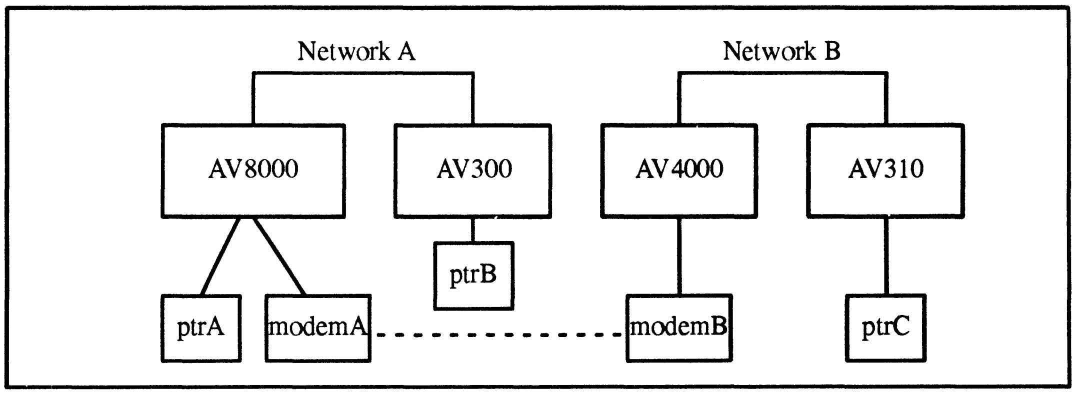

Figure 11–3 Methods of Connecting a Printer to a Computer

The AViiON 8000 system accesses printer **ptrA** through a direct connection and accesses printer **ptrB** over the network. The AViiON 300 system access printer **ptrB** via direct connection and printer **ptrA** over the network. The AViiON 4000 and AViiON 310 systems can both access printer **ptrC**, the former via the network and the latter via direct connection. The AViiON 8000 and AViiON 4000 systems have modems, meanwhile, which allows either system on either network to access printers on the other network. Modem connections are discussed in the next section.

The simplest connection method is by connecting a printer directly to your computer. You may, however, want to connect a printer to a network (so it can be shared with other computers or workstations), or to a modem. Whichever method you use, you must describe it to the LP print service after you have connected the hardware.

To define the connection method for a new printer for your print service, invoke the **lpadmin** command, specifying a connection method through either the **–v** option for a directly connected printer or the **–U** option for a printer directly connected to a network or a printer connected to a modem.

Direct Connections

The simplest and most common method by which printers are connected to a computer is direct connection. If you use this method, you generally need to specify only two items on the command line when you make the connection: the name of the printer and the name of the connecting port. To connect a printer directly to your computer enter the following command:

```
# lpadmin -p printer-name -v pathname ↵
```

where *pathname* is the name of the special device file representing the printer port. The following are examples of typical names of special device files.

```
/dev/tty00  (serial)
/dev/tty16  (serial)
/dev/lp  (parallel)
```

(For details about using these files, see "Printer Port Characteristics.")

Using a Printer As a Login Terminal
Some directly connected printers can also be used as terminals for login sessions. If you want to use a printer as a terminal, you must arrange for the LP print service to handle it as such. To do so, use the **–l** option to the **lpadmin** command, as follows:

```
# lpadmin -p printer-name -v pathname -l ↵
```

As before, *pathname* is the name of the special file representing the printer port. If the **–l** option is specified, the printer will be disabled automatically whenever the LP print service is started, and therefore will have to be manually enabled before it can be used for printing. For instructions on manually enabling a printer, see "Enabling and Disabling a Printer" (under "Making Printers Available") later in this section.

Connections to Networks and Modems

In an environment where a printer is located is so far from the computer that a direct connection is not possible or practical, you can use a modem or network to access the printer. For example, you might have one printer in use with a single terminal at a branch office located a few miles from your main site, or you may want to share a printer with computers that are not on a common network.

The LP print service establishes a connection to the printer as necessary to print requests; at the end of each request the connection is dropped, making the printer available to the next system that calls it. Thus the printer gets shared by the users of all the computers, more or less equally.

There are two methods for connecting printers that are not directly connected to your system: attached directly to a network and through a dial-up modem. The LP print service uses UUCP to handle both methods.

When a modem connection is used, the printer must be connected to a dialed modem, and the dial-out modem must be connected to the computer. Whether printers are connected to a modem or directly to a network, the connection must be described to UUCP. For instructions on describing either type of connection, see the chapter on network management.

To make a printer connected in one of these ways available to your users, enter the following command:

```
# lpadmin -p printer-name -U dial-info ↵
```

where *dial–info* is either the telephone number to be dialed to reach the printer's modem or the system name entered in the UUCP **Systems** database for the printer. The **–U** option provides a way to link a single printer to your print service. It does not allow you to connect with a print service on another system.

A note on printers connected to a modem or directly to a network: if the printer or port is busy, the LP print service will automatically retry later. This retry rate is 10 minutes if the printer is busy, and 20 minutes if the port is busy. These rates are not adjustable; however, you can force an immediate retry by issuing the **enable** command for the printer. If the port or printer is likely to be busy for an extended period, you should issue the **disable** command.

The **lpstat –p** command reports the reason for a failed dial attempt. Also, if you are alerted to a dialing fault (see "Printer Fault Alerting," later in this section), the alert message will give the reason for the fault. These messages are identical to the error messages produced by UUCP for similar problems. See Appendix A for a list of UUCP error messages and explanations.

In summary, to add printers to your system, use the **lpadmin** command, specifying a connection method through one of two options: the **–v** option for a directly connected printer or the **–U** option for a networked printer.

System Name

This section does not apply if you are making only a local printer accessible to users on your system. A remote printer is one that is connected to a system other than your local system that you can access only via that remote system.

There are some exceptional cases, however. For example, if only one of the printers has a particular typesetter needed for some print jobs, then users from many systems will want to access it from time to time.

Licensed Material – Property of copyright holder(s)

Alternatively, a large community of users on a local area network may want to pool all printers on a single system, where they can share them. When this is done, the system supporting the printers becomes a printer server.

To make accessible a printer that is remote, the name of the system on which the printer resides must be registered with the print service. If the remote printer resides on a DG/UX Release 5.4 system or on a System V Release 4 system, enter:

```
# lpsystem system-name ↵
```

If the remote printer resides on a BSD system, enter

```
# lpsystem -t bsd system-name ↵
```

For details about the options available with this command, see the **lpsystem**(1M) manual page.

In either case, after entering the **lpsystem** command, enter the **lpadmin** command, as follows:

```
# lpadmin -p printer -s system-name ↵
```

where *printer* is the name by which your users identify the remote printer. You can usually use the same name used for that printer by the remote system. If the name used by the remote system is the same name used for an existing printer or class on your system, you must use a different name. To assign a different name to a remote printer, enter the following:

```
# lpadmin -p local-name -s system-name!remote-name ↵
```

For example, imagine you want your users to have access to a printer called **psjet2** that resides on a remote system called **newyork**. Because you already have a printer called **psjet2** on your own system, you want to give the remote printer a new name on your system: **psjet3**. Request the new name by entering the following:

```
# lpadmin -p psjet3 -s newyork!psjet2 ↵
```

Before you add a remote printer to your system, be sure communications between your system and the network have been properly set up and verified.

Allowing Remote Users to Access Local Printers

Making the printers on your local system accessible to users on remote systems is a two-step process: you must configure the port monitor on the local system, and you must register the remote system with the local LP print service. This section provides instructions for these tasks.

Configuring the Local Port Monitor

If a remote system requires access to printers connected directly to your system, you need to configure the local port monitor for the network you share to accept service requests and to notify the LP print service of such requests. For DG/UX Release 5.4 systems or System V systems calling your systems, issue the following command:

```
# pmadm -a -p netname -s lp -i root -v 'nlsadmin -V' -m 'nlsadmin \
    -o /var/spool/lp/fifos/listenS5' ↵
```

where *netname* is the name of a network such as **tcp**.

If you expect users on BSD systems to send print requests to your system, then you need to configure your local port monitor. First, however, you need to know the Internet address of your system. To get this address in the correct hexadecimal format, run the **lpsystem** command with the **–A** option, as follows:

```
# lpsystem –A ↵
```

You then execute the following command, substituting the hexadecimal number for the argument address:

```
# pmadm –a –p tcp –s lpd –i root –v 'nlsadmin –V' –m 'nlsadmin \
     –o /var/spool/lp/fifos/listenBSD –A'xaddress'' ↵
```

Adding a System Entry
If you want your system to accept jobs from a remote system (and vice-versa), the print service must know about that system. The **lpsystem** command allows you to register remote systems with the local print service. Run the command as follows:

```
# lpsystem –s system–name ↵
```

where *system–name* is the name of the remote system.

If the remote host uses a pre-DG/UX Release 5.4 LP scheduler, you also need to add the host's Internet address to your **hosts** database. To add an entry to your **hosts** database, use the TCP/IP menu of **sysadm**'s Networking Menu.

Interface Program

This section does not apply if you are making a remote printer accessible to users on your system. This is the program the LP print service uses to manage the printer each time a file is printed. It has several tasks:

- to initialize the printer port (the connection between the computer and the printer)
- to initialize the printer (restore it to a normal state in case a previously printed file has left it in an unusual state) and set the character pitch, line pitch, page size, and character set requested by the user
- to print a banner page
- to run a filter that prepares the file for printing
- to manage printer faults

If you do not choose an interface program, the standard one provided with the LP print service will be used. This should be sufficient for most of your printing needs. If you prefer, however, you can change it to suit your needs, or completely rewrite your own interface program, and then specify it when you add a new printer. See "Customizing the Print Service" later in this section for details on how to customize an interface program.

If you are using the **standard** interface program, you needn't specify it when adding a printer. If, however, you will be using a different interface program on a local printer, you can refer to it either by specifying its full pathname or by referring to another printer using the same interface program.

To identify a customized interface program by name, specify the printer name and the pathname of the interface program, as follows:

```
# lpadmin -p printer-name -i pathname ↵
```

To use a customized interface program of another printer, specify the printer names as follows:

```
# lpadmin -p printer-name1 -e printer-name2 ↵
```

Printer–name1 is the name of the printer you are adding; *printer–name2* is the name of an existing printer that is using the customized interface program.

Printer Type

A printer type is the generic name for a printer. When you set up your system you can enhance its ability to serve your users by classifying, on the basis of type, the printers available through the print service. Assigning a type for each printer is also important because the LP software extracts information about printers from the **terminfo** database on the basis of type. This information includes the list of the printer's capabilities that is used to check the configuration information you supply to the print service. (By checking information provided by you against the capabilities of the printer, the print service can catch any inappropriate information you may have supplied.) The **terminfo** database also specifies the control data needed to initialize a particular printer before printing a file.

You can assign several types to a printer if the printer is capable of emulating more than one kind of printer. If you specify more than one printer type, the LP print service will use one of them as appropriate for each print request.

Although you do not need to specify a printer type, we recommend that you do so; when a printer type is specified, better print services can be provided.

To specify a printer type, use the following command line:

```
# lpadmin -p printer-name -T printer-type-list ↵
```

If you give a list of printer types, separate the names with commas. If you do not define a printer type, the default **unknown** will be used.

Content Types

While the printer type tells the LP print service what types of printers are being added, the content types tell the LP print service what types of files can be printed directly on each printer. Most printers can print files of two types: the same type as the printer type (if the printer type is defined) and the type **simple**, (meaning an ASCII file) which is the default content type for all printers.

Some printers, though, can accept (and print properly) several different types of files. When adding this kind of printer, specify the names of the content types the new printer accepts by adding these names to the list. (By default, the list contains only one type: **simple**.) If you are adding a remote printer, enter the following command on the remote system to list the content types that it accepts:

```
# lpstat -p printer -l ↵
```

To specify the list of content types for the local printer, enter the following command:

```
# lpadmin -p printer-name -I content-type-list ↵
```

The *content-type-list* is a list of names separated by commas or spaces. If you use spaces to separate the names, enclose the entire list (but not the **-I**) in quotes.

Content type names may look a lot like printer type names, but you are free to choose names that are meaningful to you and the people using the printer. (The names **simple** and **any** are recognized as having particular meanings by the LP print service; be sure to use them consistently. The name **terminfo** is also reserved, as a reference to *all* types of printers.) The names must contain no more than fourteen characters and may include only letters, digits, and underscores. Table 11–2 describes some accepted content types.

Table 11–2 Content Types

Types	Description
troff	Device independent output from **troff**
otroff	CAT typesetter instructions generated by BSD (old) **troff**
tex	DVI format files
plot	Plotting instructions for Tektronix displays and devices
raster	Raster bitmap format for Varian raster devices
cif	Output of BSD **cifpbt**
fortran	ASA carriage control format
postscript	PostScript language
simple	ASCII file

When a file is submitted to the LP print print service for printing with the printer specified by the **-d any** option of the **lp** command, the print service searches for a printer capable of handling the job. The print service can identify an appropriate printer through either the content type name or the printer type name. Therefore, you may specify either name (or no name) when submitting a file for printing. If the same content type is printable by several different types of printers, you should use the same content type names when you add those printers. This makes it easier for the people using the printers, because they can use the same name to identify the type of file they want printed regardless of the printing destination.

Most manufacturers produce printers that accept simple ASCII files. While these printers are different types (and thus have different initialization control sequences), they may all be capable of handling the same type of file, which we call **simple**. As another example, several manufacturers may produce printers that accept ANSI X3.64 defined escape sequences;

however, the printers may not support all the ANSI capabilities; they may support different sets of capabilities. You may want to differentiate them by assigning different content types to these printers.

However, while it may be desirable (in situations such as these) to list content types for each printer, it is not always necessary to do so. If you don't, the printer type will be used as the name of the content type the printer can handle. If you have not specified a printer type, the LP print service will assume the printer can print only files of content type **simple**. This may be sufficient if you require users to specify the proper printer explicitly and if files are properly prepared for the printer before being submitted for printing.

The Default Content Type: simple
Files of content type simple are assumed to contain only two types of characters, printable ASCII characters and the following control characters:

backspace	moves the carriage back one space, except at the beginning of a line.
tab	moves the carriage to the next tab stop; by default, stops are spaced every 8 columns on most printers.
linefeed	moves the carriage to the beginning of the next line (may require special port settings for some printers—see "Printer Port Characteristics" below).
form feed	moves the carriage to the beginning of the next page.
carriage return	moves the carriage to the beginning of the same line (may fail on some printers).

The word "carriage" may be archaic for modern laser printers, but these printers do actions similar to those done by a carriage. If a printer can handle several types of files, including **simple**, you must include simple in the content type list; the type **simple** is not automatically added to any list you give. If you *don't* want a printer to accept files of type **simple**, give a blank *content-type-list*, as follows:

```
# lpadmin -p printer-name -I "" ↵
```

Printer Port Characteristics

This section applies only to local printers.

The interface program needs to set the characteristics of the port to which the printer is connected. These characteristics define the low level communications with the printer. Included are the baud rate; use of XON/XOFF flow control; 7, 8, or other bits per byte; type of parity; and output post-processing. The standard interface program will use the **stty** command to initialize the printer port, minimally setting the baud rate and a few other default characteristics.

The default characteristics applied by the standard interface program are listed below.

Default	Meaning
9600	9600 baud rate
cs8	8-bit bytes

–cstopb	1 stop bit per byte
–parenb	no parity generation
ixon	enable XON/XOFF flow control
–ixany	allow only XON to restart output
opost	post-process data stream as listed below:

–olcuc	don't map lower-case to upper-case
onlcr	map linefeed into carriage-return/linefeed
–ocrnl	don't map carriage-return into linefeed
–onocr	output carriage-returns even at column 0
nl0	no delay after linefeed
cr0	no delay after carriage-return
tab0	no delay after tab
bs0	no delay after backspace
vt0	no delay after vertical tab
ff0	no delay after form-feed

You may find that the default characteristics are sufficient for your printers; however, printers vary enough that you are likely to find that you have to set different characteristics. For a complete list of characteristics, see the **stty**(1) manual page.

If you have a printer that requires printer port characteristics other than those handled by the **stty** program, you will have to customize the interface program. See "Customizing the Print Service" for help.

When you add a new printer, you may specify an additional list of port characteristics. The list you give will be applied after the default list, so that you do not need to include in your list items that you don't want to change. Specify the additional list as follows:

```
# lpadmin –p printer–name –o "stty='stty–option–list'" ↵
```

Note that both the double quotes and single quotes are needed if you give more than one item in the *stty–option–list.*

As one example, suppose your printer is to be used for printing graphical data, where linefeed characters should be output alone, without an added carriage-return. You would enter the following command:

```
# lpadmin –p printer–name –o "stty=–onlcr" ↵
```

Note that the single quotes are omitted because there's only one item in the list.

As another example, suppose your printer requires odd parity for data sent to it. You would enter the following command:

```
# lpadmin -p printer-name -o "stty='parenb parodd cs7'" ⤶
```

Character Sets or Print Wheels

Although your users may use character sets or print wheels that have been mounted on a remote printer (by the administrator of the remote system on which that printer resides), you cannot mount a character set or a print wheel on a remote printer. Printers differ in the way they can print in different font styles. Some have changeable print wheels, some have changeable font cartridges, others have preprogrammed, selectable character sets.

When adding a printer, you may specify what print wheels, font cartridges, or character sets are available with the printer. If you're adding a remote printer and you want your users to be able to use character sets or print wheels that have been mounted by the administrator of the remote system, you must list those character sets and print wheels, just as you would list the character sets and print wheels on a local printer. (See instructions below.)

Only one of these is assumed to apply to each printer. From the point of view of the LP print service, however, print wheels and changeable font cartridges are the same because they require you to intervene and mount a new print wheel or font cartridge. Thus, for ease of discussion, only print wheels and character sets will be mentioned.

When you list the print wheels or character sets available, you will be assigning names to them. These names are for your convenience and the convenience of the users. Because different printers may have similar print wheels or character sets, you should use common names for all printers. This allows a user to submit a file for printing and ask for a particular font style, without regard for which printer will be used or whether a print wheel or selectable character set is used.

If the printer has mountable print wheels, you need only list their names. If the printer has selectable character sets, you need to list their names and map each one into a name or number that uniquely identifies it in the **terminfo** database. Use the following command to determine the names of the character sets listed in the **terminfo** database.

```
# tput -T printer-type csnm 0 ⤶
```

Printer–type is the name of the printer type in question. The name of the 0th character set (the character set obtained by default after the printer is initialized) will be printed. Repeat the command, using **1**, **2**, **3**, and so on in place of the **0**, to see the names of the other character sets. In general, the **terminfo** names should closely match the names used in the user documentation for the printer; however, because not all manufacturers use the same names, the **terminfo** names may differ from one printer type to the next. For the LP print service to be able to find the names in the **terminfo** database, you must specify a printer type. See "Printer Type" above.

To specify a list of print wheel names when adding a printer, enter the following command.

```
# lpadmin -p printer-name -S print-wheel-list ⤶
```

The *print–wheel–list* is a comma or space separated list of names. If you use spaces to separate the names, enclose the entire list (but not the **–S**) in quotes.

To specify a list of character set names and to map them into **terminfo** names or numbers, enter the following command:

```
# lpadmin -p printer-name -S character-set-list ↵
```

The *character-set-list* is also a comma or space separated list; however, each item in the list looks like one of the following:

```
csN=character-set-name
character-set-name1=character-set-name2
```

The *N* in the first case is a number from 0 to 63 that identifies the number of the character set in the **terminfo** database. The *character-set-name*1 in the second case identifies the character set by its **terminfo** name. In either case the name to the right of the equal sign (=) is the name you may use as an alias of the character set. You do not have to provide a list of aliases for the character sets if the **terminfo** names are adequate. You may refer to a character set by number, by **terminfo** name, or by your alias.

For example, suppose your printer has two selectable character sets (sets #1 and #2) in addition to the standard character set (set #0). The printer type is **5310**. You enter the following commands to determine the names of the selectable character sets.

```
% tput -T 5310 csnm 1 ↵
english
% tput -T 5310 csnm 2 ↵
finnish
```

The words `english` and `finnish`, the output of the commands, are the names of the selectable character sets. You feel that the name `finnish` is adequate for referring to character set #2, but better names are needed for the standard set (set #0) and set #1. You enter the following command to define synonyms.

```
# lpadmin -p printer-name -S "cs0=american, english=british" ↵
```

The following three commands will then produce identical results.

```
# lp -S cs1 -d any ... ↵

# lp -S english -d any ... ↵

# lp -S british -d any ... ↵
```

If you do not list the print wheels or character sets that can be used with a printer, then the LP print service will assume the following: a printer that takes print wheels has only a single, fixed print wheel, and users may not ask for a special print wheel when using the printer; and a printer that has selectable character sets can take any **cs***N* name or **terminfo** name known for the printer.

Alerting to Mount a Print Wheel

This section does not apply if you are making a remote printer available to users on your system.

If you have printers that can take changeable print wheels, and have listed the print wheels allowed on each, then users will be able to submit a print request to use a particular print wheel.

Until it is mounted though (see "Mounting a Form or Print Wheel" in this section), a request for a print wheel will stay queued and will not be printed. You could periodically monitor the number of print requests pending for a particular print wheel, but the LP print service provides an easier way: you can ask to be alerted when the number of requests waiting for a print wheel has exceeded a specified threshold.

You can choose one of several ways to receive an alert.

- You can receive an alert via electronic mail. For more information on the mail system, see the **mailx**(1) manual page.
- You can receive an alert written to any terminal on which you are logged in. See the manual page for the **write**(1) command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

If you elect to receive no alerts, you are responsible for checking to see whether any print requests haven't printed because the proper print wheel isn't mounted. In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the print wheel is mounted. You can choose the rate of repeated alerts, or you can opt to receive only one alert for each print wheel.

To arrange for alerting to the need to mount a print wheel or character set, enter one of the following commands:

```
# lpadmin -S print-wheel-names -A mail -Q requests -W minutes ↵
# lpadmin -S print-wheel-names -A write -Q requests -W minutes ↵
# lpadmin -S print-wheel-names -A 'command' -Q requests -W minutes ↵
```

The `print-wheel-names` argument may be a space- or comma-separated list of print wheels or character sets. The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run the *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and current directory. The argument *requests* is the number of requests that need to be waiting for the print wheel before the alert is triggered, and the argument *minutes* is the number of minutes between repeated alerts.

If you do not want the print service to issue an alert when a print wheel needs to be mounted, enter the following:

```
# lpadmin -S print-wheel-names -A none ↵
```

If you want mail sent or a message written to another user when a printer fault occurs, use the third command with the option **-A'mail** *username*' or **-A'write** *username*'.

When you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts (for the current case only), by executing the following command.

```
# lpadmin -S print-wheel-names -A quiet ↵
```

Once the print wheel has been mounted and unmounted again, alerts will start again if too many requests are waiting. Alerts will also start again if the number of requests waiting falls below the **–Q** threshold and then rises up to the **–Q** threshold again, as when waiting requests are canceled, or if the type of alerting is changed.

If *print–wheel–name* is **all** in any of the commands above, the alerting condition will apply to all print wheels for which an alert has already been defined.

If you don't define an alert method for a print wheel, you will not receive an alert to mount it. If you do define a method without the **–W** option, you will be alerted once for each occasion.

Forms Allowed

For information about how to define, mount, and set up alerting to mount a form, see "Providing Forms" later in this section.

You can control the use of preprinted forms on any printer, including remote printers. (Although you cannot mount forms on remote printers, your users may use forms on remote printers.) You may want to do this, for instance, if a printer is not well suited for printing on a particular form because of low print quality, or if the form cannot be lined up properly in a local printer.

The LP print service will use a list of forms allowed or denied on a printer to warn you against mounting a form that is not allowed on the printer. However, you have the final word on this; the LP print service will not reject the mounting. The LP print service will, however, reject a user's request to print a file on a printer using a form not allowed on that printer. If, however, the printer is a local printer and the requested form is already mounted, the request will be printed on that form.

If you try to allow a form for a printer, but the printer does not have sufficient capabilities to handle the form, the command will be rejected.

The method of listing the forms allowed or denied for a printer is similar to the method used to list users allowed or denied access to the **cron** and **at** facilities. See the manual page for **cron**(1M) and **crontab**(1). Briefly, the rules are as follows:

- An allow list is a list of forms that are allowed to be used on the printer. A deny list is a list of forms that are not allowed to be used on the printer.
- If the allow list is not empty, only the forms listed are allowed; the deny list is ignored. If the allow list is empty, the forms listed in the deny list are not allowed. If both lists are empty, there are no restrictions on which forms may be used other than those restrictions that apply to a printer of a particular type, such as a PostScript printer for which a license is required.
- Specifying **all** in the allow list allows all forms; specifying **all** in the deny list denies all forms.

You can add names of forms to either list using one of the following commands:

```
# lpadmin -p printer-name -f allow:form-list ↵
# lpadmin -p printer-name -f deny:form-list ↵
```

The *form–list* is a comma or space separated list of names of forms. If you use spaces to separate names, enclose the entire list (including the **allow:** or **deny:** but not the **–f**) in quotes.

The first command shown above adds names to the allow list and removes them from the deny list. The second command adds names to the deny list and removes them from the allow list. To make the use of all forms permissible, specify **allow:all**; to deny permission for all forms, specify **deny:all.**

If you do not use this option, the LP print service will consider that the printer denies the use of all forms. It will, however, allow you to mount any form, thereby making it implicitly available to use. (See "Mounting a Form or Print Wheel" later in this section for more information.)

Printer Fault Alerting

This section does not apply if you are making a remote printer accessible to users on your system. The LP print service provides a framework for detecting printer faults and alerting you to them. Faults can range from simple problems, such as running out of paper or ribbon, or needing to replace the toner, to more serious faults, such as a local power failure or a printer failure. The range of fault indicators is also broad, ranging from dropping carrier (the signal that indicates that the printer is on line), to sending an XOFF, to sending a message. Only two classes of printer fault indicators are recognized by the LP print service itself: a drop in carrier and an XOFF not followed in reasonable time by an XON. However, you can add filters that recognize any other printer fault indicators, and rely on the LP print service to alert you to a fault when the filter detects it. For a description of how to add a filter, see "Providing Filters" in this section. For a description of how a filter should let the LP print service know a fault has occurred, see "Customizing the Print Service" in this section.

You can choose one of several ways to receive an alert to a printer fault:

- You can receive an alert via electronic mail. See the manual page for the **mailx**(1) command.
- You can receive an alert written to any terminal on which you are logged in. See the manual page for the **write**(1) command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

If you elect to receive no alerts, you will need a way of finding out about the faults and fixing them; the LP print service will not continue to use a printer that has a fault. In addition to the method of alerting, you can also arrange for repeated alerts every few minutes until the fault is cleared. You can choose the rate of repeated alerts, or you can choose to receive only one alert per fault. Without a filter that provides better fault detection, the LP print service cannot automatically determine when a fault has been cleared except by trying to print another file. It will assume that a fault has been cleared when it is successfully able to print a file. Until that time, if you have asked for only one alert per fault, you will not receive another alert. If, after you have fixed a fault, but before the LP print service has tried printing another file, the printer faults again, or if your attempt to fix the fault fails, you will not be notified. Receiving repeated alerts per fault, or requiring manual re-enabling of the printer (see "Printer Fault Recovery," below), will overcome this problem.

To arrange for alerting to a printer fault, enter one of the following commands:

```
# lpadmin -p printer-name -A mail -W minutes ↵
# lpadmin -p printer-name -A write -W minutes ↵
# lpadmin -p printer-name -A'command' -W minutes ↵
```

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run the *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*. The environment includes environment variables, user and group IDs, and current directory. The *minutes* argument is the number of minutes between repeated alerts.

If you do not want the LP print service to issue an alert when a fault occurs, enter the following:

```
# lpadmin -p printer-name -A none ↵
```

If you want mail sent or a message written to another user when a printer fault occurs, use the third command with the option **–A'mail** *username*' or **–A'write** *username*'.

Once a fault occurs and you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts (for the current fault only), by executing the following command:

```
# lpadmin -p printer-name -A quiet ↵
```

If the *printer–name* is **all** in any of the commands above, the alerting condition will apply to all printers.

If you don't define an alert method, you will receive mail once for each printer fault. If you define a method without the **–W** option, you will be alerted once for each fault.

Printer Fault Recovery

This section does not apply if you are making a remote printer accessible to users on your system. When a printer fault has been fixed and the printer is ready for printing again, the LP print service will recover in one of three ways:

- It will continue printing at the top of the page where printing stopped.
- It will restart printing at the beginning of the print request that was active when the fault occurred.
- It will wait for you to tell the LP print service to re-enable the printer.

The ability to continue printing at the top of the page where printing stopped requires the use of a filter that can wait for a printer fault to be cleared before resuming properly. Such a filter probably has to have detailed knowledge of the control sequences used by the printer so it can keep track of page boundaries and know where in a file printing stopped. The default filter used by the LP print service cannot do this. If a proper filter is not being used, you will be notified in an alert if recovery cannot proceed as you want.

To specify the way the LP print service should recover after a fault has been cleared, enter one of the following commands:

```
# lpadmin -p printer-name -F continue ↵
# lpadmin -p printer-name -F beginning ↵
# lpadmin -p printer-name -F wait ↵
```

These commands direct the LP print service, respectively, to continue at the top of the page, restart from the beginning, or wait for you to enter an **enable** command to re-enable the printer (see "Enabling and Disabling Printer" for information on the **enable** command).

If you do not specify how the LP print service is to resume after a printer fault, it will try to continue at the top of the page where printing stopped, or, failing that, at the beginning of the print request.

If the recovery is **continue**, but the interface program does not stay running so that it can detect when the printer fault has been cleared, printing will be attempted every few minutes until it succeeds. You can force the LP print service to retry immediately by issuing an **enable** command.

User Access Restrictions

You can control which users are allowed to use a particular printer on your system. For instance, if you're designating one printer to handle sensitive information, you don't want all users to be able to use the printer. Another time you might want to do this is when you're authorizing the use of a high quality printer which produces expensive output.

The LP print service will use a list of users allowed or denied access to a printer. The LP print service will reject a user's request to print a file on a printer he or she is not allowed to use. If your users have access to remote printers, or if users on other systems have access to printers on your system, make sure that the allow and deny lists for those printers on your computer match the allow and deny lists on the remote system where the remote printers reside. If these two sets of lists don't match, your users may receive conflicting messages (some accepting jobs, and others refusing jobs) when submitting requests to remote printers.

The method of listing the users allowed or denied access to a printer is similar to the method used to list users allowed or denied access to the **cron** and **at** facilities, and the method described above in "Forms Allowed." Briefly, the rules are as follows:

- An allow list is a list of users allowed to use the printer. A deny list is a list of users denied access to the printer.
- If the allow list is not empty, only the users listed are allowed; the deny list is ignored. If the allow list is empty, users listed in the deny list are not allowed. If both lists are empty, there are no restrictions on who may use the printer.
- Specifying **all** in the allow list allows everybody access to the printer; specifying **all** in the deny list denies access to everybody except the user **lp** and the super-user (**root** or **sysadm**).

You can add names of users to either list using one of the following commands:

```
# lpadmin -p printer-name -u allow:user-list ↵
# lpadmin -p printer-name -u deny:user-list ↵
```

The *user–list* is a comma or space separated list of names of users. If you use spaces to separate the names, enclose the entire list (including the **allow:** or **deny:** but not the **–u**) in quotes. Each item in the *user–list* may take any of the following forms:

user	*user* on any system
all	All users on all systems
system!*user*	*user* on *system* only
!*user*	*user* on local system only
all!*user*	*user* on any system
all!all	All users on all systems
system!**all**	All users on *system*
!**all**	All users on local system

The first command shown above adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list.

If you do not use this option, the LP print service will assume that everybody may use the printer.

Inclusion of Banner Page in Output

Most users want to have the output of each print request preceded by a banner page. A banner page shows who requested the printing, the request ID for it, and when the output was printed. It also allows for an optional title that the requester can use to better identify a printout. Finally, the banner page greatly eases the task of separating a sequence of print requests so that each may be given to the correct user.

Sometimes a user needs to avoid printing a banner page. The likely occasions are when the printer has forms mounted that should not be wasted, such as payroll checks or accounts payable checks. Printing a banner page under such circumstances may cause problems.

Enter the following command to allow users to request no banner page:

```
# lpadmin -p printer-name -o nobanner ⤶
```

If you later change your mind, you can reverse this choice by entering the following command:

```
# lpadmin -p printer-name -o banner ⤶
```

If you do not allow a user to skip the banner page, the LP print service will reject all attempts to avoid a banner page when printing on the printer. This is the default action.

Printer Description

An easy way to give users of the LP print service helpful information about a printer is by adding a description of it. This description can contain any message you like, including the

Licensed Material – Property of copyright holder(s)

number of the room where the printer is found, the name of the person to call with printer problems, and so forth.

Users can see the message when they use the **lpstat–D–p***printer–name* command.

To add a description of a printer, enter the following command.

```
# lpadmin –p printer–name –D 'text' ↵
```

The *text* is the message. You'll need to include the quotes if the message contains blanks or other characters that the shell might interpret if the quotes are left out.

Default Printing Attributes

The attributes of a printing job include the page size, print spacing (character pitch and line pitch), and **stty** options for that job. If a user requests a job to be printed on a particular form, the printing attributes defined for that form will be used for that job. When, however, a user submits a print request without requesting a form, the print service uses one of several sets of default attributes.

- If the user has specified attributes to be used, those attributes will take precedence.
- If the user has not specified attributes, but the administrator has executed the **lpadmin –o** command, the default attributes for that command will take precedence.
- If neither of the above, the attributes defined in the **terminfo** database for the printer being used will take precedence.

The LP print service lets you override the defaults for each printer. Doing so can make it easier to submit print requests by allowing you to designate different printers as having different default page sizes or print spacing. A user can then simply route a file to the appropriate printer to get a desired style of output. For example, you can have one printer dedicated to printing wide (132-column) output, another printing normal (80-column by 66-line) output, and yet another printing letter quality (12 characters per inch, 8 lines per inch) output.

You can independently specify four default settings, page width, page length, character pitch, and line pitch. You can scale these to fit your needs: the first two can be given in characters and lines, or inches or centimeters. The last two can be given as characters and lines per inch or per centimeter. In addition, the character pitch can be specified as **pica** for 10 characters per inch (cpi), **elite** for 12 cpi, or **compressed** for the maximum cpi the printer can provide (up to a limit of 30 cpi).

Set the defaults using one or more of the following commands:

```
# lpadmin –p printer–name –o width=scaled–number ↵
# lpadmin –p printer–name –o length=scaled–number ↵
# lpadmin –p printer–name –o cpi=scaled–number ↵
# lpadmin –p printer–name –o lpi=scaled–number ↵
```

Append the letter **i** to the *scaled–number* to indicate inches, or the letter **c** to indicate centimeters. The letter **i** for character pitch (**cpi**) or line pitch (**lpi**) is redundant. You can also give **pica**, **elite**, or **compressed** instead of a number for the character pitch.

If you don't provide defaults when you configure a printer, then the page size and print spacing will be taken from the data for your printer type in the **terminfo** database. (If you do not specify a printer type, the type will be **unknown**, for which there is an entry in the **terminfo** database.) You can find out what the defaults will be by first defining the printer configuration without providing your own defaults, then using the **lpstat** command to display the printer configuration. The command

```
# lpstat -p printer-name -l ↵
```

will report the default page size and print spacing.

Printer Class Membership

This section does not apply if you are making a remote printer accessible to users on your system. It is occasionally convenient to treat a collection of printers as a single class. The benefit is that a user can submit a file for printing by a member of a class, and the LP print service will pick the first printer in the class that it finds free. This allows faster turn-around, as printers are kept as busy as possible.

Classes aren't needed if the only purpose is to allow a user to submit a print request by type of printer. The **lp -T** *content-type* command allows a user to submit a file and specify its type. The first available printer that can handle the type of the file will be used to print it. The LP print service will avoid using a filter, if possible, by choosing a printer that can print the file directly over one that would need it filtered first. See "Providing Filters" for more information about filters.

Classes do have uses, however. One use is to put into a class a series of printers that should be used in a particular order. If you have a high speed printer and a low speed printer, for instance, you probably want the high speed printer to handle as many print requests as possible, with the low speed printer reserved for use when the other is busy. Because the LP print service always checks for an available printer in the order the printers were added to a class, you could add the high speed printer to the class before the low speed printer, and let the LP print service route print requests in the order you wanted.

Until you add a printer to a class, it doesn't belong to one. If you want to do so, use the following command:

```
# lpadmin -p printer-name -c class-name ↵
```

If the class *class–name* doesn't exist yet, it will be created. If you want to remove a printer from a class without deleting the printer, enter the following command:

```
# lpadmin -p printer-name -r class-name ↵
```

The class name may contain a maximum of fourteen alphanumeric characters and underscores. Class names and printer names must be unique. Because they are, a user can specify the destination for a print request without having to know whether it's a class of printers or a single printer.

System Default Destination

You can define the printer or class to be used to print a file when the user has not explicitly asked for a particular destination and has not set the **LPDEST** shell variable. The printer or class must already exist.

Make a printer or class the default destination by entering the following command:

```
# lpadmin -d printer-or-class-name ↵
```

If you later decide that there should be no default destination, enter a null *printer–or–class–name* as in the following command:

```
# lpadmin -d ↵
```

If you don't set a default destination, there will be none. Users will have to explicitly name a printer or class in each print request, (unless they specify the **–T** *content–type* option) or will have to set the LPDEST shell variable with the name of a destination.

Mounting a Form or Print Wheel

See "Providing Forms" later in this section for information about pre-printed forms. Before the LP print service can start printing files that need a preprinted form or print wheel, you must physically mount the form or print wheel on a printer, and notify the LP print service that you have mounted it. (It is not necessary for a form to be included on the allow list in order to mount it.) If alerting has been set on the form or print wheel, you will be alerted when enough print requests are queued waiting for it to be mounted. (See "Alerting to Mount a Form" below and "Alerting to Mount a Print Wheel" above.)

When you mount a form you may want to see if it is lined up properly. If an alignment pattern has been defined for the form, you can ask that this be repeatedly printed after you've mounted the form, until you have adjusted the printer so that the alignment is correct.

Mounting a form or print wheel involves first loading it onto the printer and then telling the LP print service that it is mounted. Because it is difficult to do this on a printer that's currently printing, and because the LP print service will continue to print files not needing the form on the printer, you will probably have to disable the printer first. Thus, the proper procedure is to follow these three steps:

1. Disable the printer, using the **disable** command.
2. Mount the new form or print wheel as described immediately after this list.
3. Re-enable the printer, using the **enable** command. (The **disable** and **enable** commands are described in "Enabling and Disabling a Printer.")

First, physically load the new form or print wheel into the printer. Then enter the following command to tell the LP print service it has been mounted.

Enter the following command if you are mounting a form:

```
# lpadmin -p printer-name -M -f form-name -a -o filebreak ↵
```

Enter the following command if you are mounting a print wheel:

```
# lpadmin -p printer-name -M -S print-wheel-name ↵
```

If you are mounting a form with an alignment pattern defined for it, you will be asked to press the Enter key before each copy of the alignment pattern is printed. After the pattern is printed, you can adjust the printer and press the Enter key again. If no alignment pattern has been defined, you won't be asked to press the Enter key. You can drop the **–a** and **–o filebreak** options if you don't want to bother with the alignment pattern.

The **–o filebreak** option tells the LP print service to add a form-feed after each copy of the alignment pattern. The actual control sequence used for the form-feed depends on the printer involved and is obtained from the **terminfo** database. If the alignment pattern already includes a form-feed, leave out the **–o filebreak** option.

To unmount a form, use the following command:

```
# lpadmin -p printer-name -M -f none  ↵
```

To unmount a print wheel, use the following command:

```
# lpadmin -p printer-name -M -S none  ↵
```

Until you've mounted a form on a printer, only print requests that don't require a form will be printed. Likewise, until you've mounted a print wheel on a printer, only print requests that don't require a particular print wheel will be printed. Print requests that do require a particular form or print wheel will be held in a queue until the form or print wheel is mounted.

Removing a Printer or Class

You can remove a printer or class if it has no pending print requests. If there are pending requests, you have to first move them to another printer or class (using the **lpmove** command), or cancel them (using the **cancel** command).

Removing the last remaining printer of a class automatically removes the class as well. The removal of a class, however, does not cause the removal of printers that were members of the class. If the printer or class removed is also the system default destination, the system will no longer have a default destination.

To remove a printer or class, enter the following command:

```
# lpadmin -x printer-or-class-name ↵
```

If all you want to do is to remove a printer from a class without deleting that printer, enter the following command:

```
# lpadmin -p printer-name -r class-name ↵
```

Putting It All Together

It is possible to add a new printer by completing a number of separate steps, shown in the commands described above. You may find it easier, however, to enter one or two commands that combine all the necessary arguments. Below are some examples.

Example 1

Add a new printer called **lp1** (of the type **455**) on printer port **/dev/tty13**. It should use the standard interface program, with the default page size of 90 columns by 71 lines, and linefeeds should *not* be mapped into carriage return/linefeed pairs.

```
# lpadmin -p lp1 -v /dev/tty13 -T 455 -o "width=90 length=71 stty=-onlcr" ↵
```

Example 2

Add a new printer called **laser** on printer port **/dev/tty41**. It should use a customized interface program, located in the directory **/usr/doceng/laser_intface**. It can handle three file types—**i10, i300**, and **impress**—and it may be used only by the users **doceng** and **docpub**. (The following command line is broken over multiple lines for readability.)

```
# lpadmin -p laser -v /dev/tty41 -i /usr/doceng/laser_intface \
      -I "i10,i300,impress" -u "allow:doceng,docpub" ↵
```

Example 3

When you added the **lp1** printer in the first example, you did not set the alerting. Do this now: have the LP print service alert you—by writing to the terminal on which you are logged in—every 10 minutes after a fault until you fix the problem.

```
# lpadmin -p lp1 -A write -W 10 ↵
```

Examining a Printer Configuration

Once you've defined a printer configuration, you probably want to review it to see if it is correct. If after examining the configuration you find you've made a mistake, just reenter the command that applies to the part that's wrong.

Use the **lpstat** command to examine both the configuration and the current status of a printer. The short form of this command gives just the status; you can use it to see if the printer exists and if it is busy, idle, or disabled. The long form of the command gives a complete configuration listing.

Enter one of the following commands to examine a printer.

```
# lpstat -p printer-name ↵
# lpstat -p printer-name -l ↵
```

(The second command is the long form.) With either command you should see one of the following lines of output.

```
printer printer-name now printing  request-id. enabled since date.

printer printer-name is idle. enabled since date.

printer printer-name disabled since date.
        reason

printer printer-name waiting for auto-retry.
        reason
```

The message `waiting for auto-retry` shows that the LP print service failed in trying to use the printer (because of the *reason* shown), and that it will try again later.

With the long form of the command, you should also see the following output:

```
Form mounted: form-name
Content types: content-type-list
Printer type: printer-type
Description: comment
Connection: connection-info
Interface: pathname
On fault: alert-method
After fault: fault-recovery
Users allowed:
     user-list
Forms allowed:
     form-list
Banner required
Character sets:
     character-set-list
Default pitch: integer CPI, integer LPI
Default page size: scaled-decimal wide, scaled-decimal long
Default port settings: stty-option-list
```

Making Printers Available

There are two steps in making a printer ready for use after you've defined the printer configuration. First, you must instruct the LP print service to accept print requests for the new printer. To do this, run the **accept** command. Second, you must activate or enable the new printer. To do this, run the **enable** command. These tasks are separate steps because you may have occasion to want to do one but not the other.

Accepting Print Requests for a New Printer

Telling the LP print service to accept print requests for the new printer is done with the **accept** command. You will read more about this command in "Managing the Printing Load," later in the section. For now, all you need to know is that you should enter the following command to let this printer be used.

```
# accept printer-or-class-name ↵
```

As you can see, this command is needed to let the LP print service start accepting print requests for a class, too. To prevent the print service from accepting any more requests, execute the following command.

```
# reject printer-or-class-name ↵
```

Enabling and Disabling a Printer

Because you may want to make sure, before printing begins, that the correct form is loaded in your printer, the correct print wheel or font cartridge is in place, and the printer is on-line, the LP print service will wait for an explicit signal from you before it starts printing files. Once you have verified that all the necessary components are in place, you can request the beginning of printing by issuing the **enable** command for a particular printer, as follows:

```
# enable printer-name ⤶
```

If you want to enable several printers simultaneously, list the printers (separating the names with spaces) on the same line as the **enable** command. Don't enclose the list in quotes.

Disabling a printer stops further print requests from being printed. (It does not, however, stop the LP print service from accepting new print requests for the printer.) From time to time you may want to disable a printer. For example, you may want to interrupt a print request, or you may want to change a form or print wheel, in which case you should disable the printer first. Normally, disabling a printer also stops the request that's currently being printed, placing it back in the queue so it can be printed later. You can, however, have the LP print service wait until the current request finishes, or even cancel the request outright.

To disable a printer, enter one of the following commands:

```
# disable -r "reason"  printer-name ⤶
# disable -W -r "reason"  printer-name ⤶
# disable -c -r "reason"  printer-name ⤶
```

The first command disables the printer, stopping the currently printing request and saving it for printing later. The other commands also disable the printer, but the second one makes the LP print service wait for the current request to finish, while the third cancels the current request. The **-c** and **-W** options are not valid when the **disable** command is run to stop a remote printer because, when run for a remote printer, **disable** stops the transferring (rather than the actual printing) of print requests. The *reason* is stored and displayed whenever anyone checks the status of the printer. You can omit it (and the **-r** option) if you don't want to specify a reason.

Several printers can be disabled at once by listing their names in the same line as the **disable** command. You can only enable or disable local printers; the loading of forms, print wheels, and cartridges in a remote printer and the enabling of that printer are the responsibility of the administrator of the remote system. You can, however, enable or disable the transfer of print requests to the remote system on which a printer is located. Only individual printers can be enabled and disabled; classes cannot.

Allowing Users to Enable and Disable a Printer

You may want to make the **enable** and **disable** commands available for use by other users. This availability is useful, for instance, if you have a small organization where anyone who spots a problem with the printer should be able to disable it and fix the problem. This is *not* a good idea if you want to keep others from interfering with the proper operation of the print services.

If you want to allow others access to the **enable** and **disable** commands, use a standard DG/UX system feature called the *setuid bit*. By assigning ownership of these commands to the user **lp** (this should have been done automatically when you installed the software), and by setting the setuid bit, you can make sure that anyone will be allowed to use the **enable** and **disable** commands. Clearing the bit removes this privilege.

To allow everybody to run **enable** and **disable**, enter the following two commands:

```
# chown lp /usr/bin/enable /usr/bin/disable ↵
# chmod u+s /usr/bin/enable /usr/bin/disable ↵
```

The first command makes the user **lp** the owner of the commands; this step should be redundant, but it is safer to run the command than to skip it. The second command turns on the setuid bit.

To prevent others from running **enable** and **disable**, enter the following command:

```
# chmod u-s /usr/bin/enable /usr/bin/disable ↵
```

Troubleshooting

Here are a few suggestions of what to do if you are having difficulty getting a printer to work.

No Output (Nothing is Printed)

The printer is sitting idle; nothing happens. First, check the documentation that came with the printer to see if there is a self-test feature you can invoke; make sure the printer is working before continuing.

There are three possible explanations when you don't receive any output.

Is the Printer Connected to the Computer?

The type of connection between a computer and a printer may vary. See your printer and computer hardware documentation.

Is the Printer Enabled?

The printer must be enabled in two ways: first, the printer must be turned on and ready to receive data from the computer. Second, the LP print service must be ready to use the printer. If you receive error messages when setting up your printer, follow the "fixes" suggested in the messages. When the printer is set up, issue the commands

```
# accept printer-name ↵
# enable printer-name ↵
```

where *printer-name* is the name you assigned to the printer for the LP print service. Now submit a sample file for printing:

```
# lp -d printer-name file-name ↵
```

Is the Baud Rate Correct?

If the baud rate (the rate at which data is transmitted) is not the same for both the computer and the printer, sometimes nothing will print (see below).

Illegible Output

The printer tries printing, but the output is not what you expected; it certainly isn't readable. There are four possible explanations for this situation:

Is the Baud Rate Correct?

Usually, when the baud rate of the computer doesn't match that of the printer, you'll get some output but it will not look at all like what you submitted for printing. Random characters will appear, with an unusual mixture of special characters and unlikely spacing.

Read the documentation that came with the printer to find out what its baud rate is. It should probably be set at 9600 baud for optimum performance, but that doesn't matter for now. If it isn't set to 9600 baud, you can have the LP print service use the correct baud rate (by default it uses 9600). If the printer is connected via a parallel port, the baud rate is irrelevant.

To set a different baud rate for the LP print service, enter the following command:

```
# lpadmin -p printer-name -o stty=baud-rate ↵
```

Now submit a sample file for printing (explained earlier in this section).

Is the Parity Setting Correct?

Some printers use a "parity bit" to ensure that the data received for printing has not been garbled in transmission. The parity bit can be encoded in several ways; the computer and the printer must agree on which one to use. If they do not agree, some characters either will not be printed or will be replaced by other characters. Generally, though, the output will look approximately correct, with the spacing of "words" typical for your document and many letters in their correct place.

Check the documentation for the printer to see what the printer expects. The LP print service will not set the parity bit by default. You can change this, however, by entering one of the following commands:

```
# lpadmin -p printer-name -o stty=oddp ↵
# lpadmin -p printer-name -o stty=evenp ↵
# lpadmin -p printer-name -o stty=-parity ↵
```

The first command sets odd parity generation, the second sets even parity. The last command sets the default, no parity.

If you are also setting a baud rate other than 9600, you may combine the baud rate setting with the parity settings, as in the sample command below.

```
# lpadmin -p printer-name -o "stty='evenp 1200'" ↵
```

Are Tabs Set Correctly?

If the printer doesn't expect to receive tab characters, the output may contain the complete content of the file, but the text may appear in a chaotic looking format, jammed up against the right margin (see below).

Is the Printer Type Correct?

See below under "Wrong Character Set or Font."

Legible Printing, but Wrong Spacing

The output contains all of the expected text and may be readable, but the text appears in an undesirable format: double spaced, with no left margin, run together, or zigzagging down the page. These problems can be fixed by adjusting the printer settings (if possible) or by having the LP print service use settings that match those of the printer. The rest of this section provides details about solving each of these types of problems.

Double Spaced

Either the printer's tab settings are wrong or the printer is adding a linefeed after each carriage return. (The LP print service has a carriage return added to each linefeed, so the combination causes two linefeeds.) You can have the LP print service not send tabs or not add a carriage return by using the **stty -tabs** option or the **-onlcr** option, respectively.)

```
# lpadmin -p printer-name -o stty=-tabs ↵
# lpadmin -p printer-name -o stty=-onlcr ↵
```

No Left Margin/Runs Together/Jammed Up

The printer's tab settings aren't correct; they should be set every 8 spaces. You can have the LP print service not send tabs by using the **-tabs** option.

```
# lpadmin -p printer-name -o stty=-tabs ↵
```

Zigzags Down the Page

The **stty onlcr** option is not set. This is set by default, but you may have cleared it accidentally.

```
# lpadmin -p printer-name -o stty=onlcr ↵
```

A Combination of Problems

If you need to use several of these options to take care of multiple problems, you can combine them in one list, as shown in the sample command below. Include any baud rate or parity settings, too.

```
# lpadmin -p printer-name -o "stty='-onlcr -tabs 2400'" ↵
```

Legible Printing but Wrong Spacing

If the wrong printer type was selected when you set up the printer with the LP print service, the wrong "control characters" can be sent to the printer. The results are unpredictable and may cause output to disappear or to be illegible, making it look like a problem described above. Another result may be that the wrong control characters cause the printer to set the wrong character set or font.

If you don't know which printer type to specify, try the following to examine the available printer types. First, if you think the printer type has a certain name, try the following command.

```
# tput -T printer-type longname ↵
```

The output of this command will appear on your terminal: a short description of the printer identified by the *printer-type*. Try the names you think might be right until you find one that identifies your printer.

If you don't know what names to try, you can examine the **terminfo** directory to see what names are available. Warning: t here are probably many names in that directory. Enter the following command to examine the directory.

```
# ls -CR /usr/share/lib/terminfo/* | more ⤶
```

Pick names from the list that match one word or number identifying your printer. Try each of the names in the other command above.

When you have the name of a printer type you think is correct, set it in the LP print service by entering the following command:

```
# lpadmin -p printer-name -T printer-type ⤶
```

Dial Out Failures

The LP print service uses UUCP to handle dial-out printers. If a dialing failure occurs and you are receiving printer fault alerts, the LP print service reports the same error reported by UUCP for similar problems. (If you haven't arranged to receive fault alerts, they are mailed, by default, to the user **lp**.) See Appendix A for a list of UUCP error messages and explanations.

Idle Printers

There are several reasons why you may find a printer idle and enabled but with print requests still queued for it:

- The print requests need to be filtered. Slow filters run one at a time to avoid overloading the system. Until a print request has been filtered (if it needs slow filtering), it will not print. Use the following command to see if the first waiting request is being filtered.

  ```
  # lpstat -o -l ⤶
  ```

- The printer has a fault. After a fault has been detected, printing resumes automatically, but not immediately. The LP print service waits about five minutes before trying again, and continues trying until a request is printed successfully. You can force a retry immediately by enabling the printer as follows:

  ```
  # enable printer-name  ⤶
  ```

- A dial-out printer is busy or doesn't answer, or all dial-out ports are busy. As with automatic continuation after a fault, the LP print service waits five minutes before trying to reach a dial-out printer again. If the dial-out printer can't be reached for an hour or two (depending on the reason), the LP print service finally alerts you to a possible problem. You can force a retry immediately by enabling the printer as follows:

  ```
  # enable printer-name  ⤶
  ```

Networking Problems

You may encounter several types of problems while trying to get files printed over a network: (1) requests being sent to remote printers may back up in the local queue; (2) requests sent to remote printers may be backed up in the remote queue; or (3) a user may receive contradictory messages about whether a remote printer has accepted a print request. The rest of this section describes each of these situations and suggests how to resolve them.

Jobs Backing Up in the Local Queue

There are a lot of jobs backing up in the local queue for a remote printer. There are three possible explanations:

- The remote system is down or the network between the local and remote systems is down. To resolve this problem, run the **reject** command for all the remote printers on your system, as follows:

  ```
  # reject printer-name ⮐
  ```

  This will stop new requests for those printers from being added to the queue. Once the system comes up again, and jobs start being taken from your queue, type **accept** *printer* to allow new jobs to be queued.

- The remote printer is disabled on the local system. The underlying DG/UX Release 5.4 network software was not set up properly. For details, see **lpsystem**(1M).

Jobs Backing Up in the Remote Queue

The remote printer has been disabled.

Conflicting Acceptance/Rejection Messages

A user enters a print request and is notified that the system has accepted it. The job is sent to a remote system and the user receives mail that the job has been rejected.

This may be happening for one of two reasons:

- the local host may be accepting requests while the remote host is rejecting them, or
- the printer's definition on the local host may not match its definition on the remote host.

The definitions of job components (such as filters, character sets, print wheels, and forms) must be identical on both the local and the remote systems if local users are to be able to access remote printers.

Providing Forms

A form is a sheet of paper, on which text or graphical displays have already been printed, that can be loaded into a local printer (that is, a printer on your system) for use in place of plain stock. Common examples of forms include company letterhead, special paper stock, invoices, blank checks, vouchers, receipts, and labels.

Typically, several copies of a blank form are loaded into a printer, either as a tray of single sheets or as a box of fan-folded paper. An application is used to generate data that will be printed on the form, thereby filling it out.

The LP print service helps you manage the use of preprinted forms, but does not provide your application any help in filling out a form; this is solely your application's responsibility. The LP print service, however, will keep track of which print requests need special forms mounted and which forms are currently mounted. It can alert you to the need to mount a new form.

This section tells you how you can manage the use of preprinted forms with the LP print service. You will see how you can

- define a new form
- change the print service's description of an existing form
- remove the print service's description of a form
- examine the print service's description of a form
- restrict user access to a form
- arrange alerting to the need to mount a form
- inform the print service that a form has been mounted

Defining a Form

When you want to provide a new form, the first thing you have to do is define its characteristics. To do so, enter information about each of the nine required characteristics (page length, page width, and so on) as input to the **lpforms** command (see below for details). The LP print service will use this information for two purposes: to initialize the printer so that printing is done properly on the form, and to send you reminders about how to handle that form. Before running the **lpforms** command, gather the following information about your new form:

Page length The length of the form, or of each page in a multi-page form. This can be expressed as the number of lines, or the size in inches or centimeters.

Page width The width of the form, expressed in characters, inches, or centimeters.

Number of pages
The number of pages in a multi-page form. The LP print service uses this number with a filter (if available) to restrict the alignment pattern to a length of one form. (See the description of alignment patterns below.) If no filter is available, the LP print service does not truncate the output.

Line pitch A measurement that shows how closely together separate lines appear on the form. It can be expressed in either lines per inch or lines per centimeter.

Character pitch A measurement that shows how closely together separate characters appear on the form. It can be expressed in either characters per inch or characters per centimeter.

Character set choice
The character set, print wheel, or font cartridge that should be used when this form is used. A user may choose a different character set for his or her own print request when using this form, or you can insist that only one character set be used.

Ribbon color If the form should always be printed using a certain color ribbon, then the LP print service can remind you which color to use when you mount the form.

Comment Any comment you wish to make about the form. This comment is available for users to see so they can understand what the form is, when it should be used, and so on.

Alignment pattern
A sample file that the LP print service uses to fill one blank form. When mounting the form, you can print this pattern on the form to align it properly. You can also define a content type for this pattern so that the printer service knows how to print it.

The LP print service does not try to mask sensitive information in an alignment pattern. If you do not want sensitive information printed on sample forms (very likely the case when you align checks, for instance) then you should mask the appropriate data. The LP print service keeps the alignment pattern stored in a safe place, where only you (that is, the user **lp** and the super-user **root** or **sysadm**) can read it.

When you've gathered this information about the form, enter it as input to the **lpforms** command. You may want to record this information first in a separate file so you can edit it before entering it with **lpforms**. You can then use the file as input instead of typing each piece of information separately after a prompt. Whichever method you use, enter the information in the following format:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set-name[,mandatory]
Ribbon color: ribbon-color
Comment:
comment
Alignment pattern: [content-type]
alignment-pattern
```

Although these attributes are described in detail on the previous page, a few points should be emphasized here. First, the phrase **mandatory** is optional and, if present, means that the user cannot override the character set choice in the form.

Second, **content-type** can be given optionally, with an alignment pattern. If this attribute is given, the print service uses the alignment pattern specified to determine, as necessary, how to filter and print the file.

With two exceptions, the information in the above list may appear in any order. The exceptions are the alignment pattern (which must always appear last) and *comment* (which must always follow the line with the **Comment:** prompt). If the *comment* contains a line beginning with a key phrase (such as **Page length**, **Page width**, and so on), precede that line with a > character so the key phrase is hidden. Be aware, though, that any initial > will be stripped from the comment when it is displayed.

Not all of the information has to be given. The defaults for the form's various attributes appear below:

Page length	66 lines
Page width	80 columns
Number of pages	1
Line pitch	6 per inch
Character pitch	10 per inch
Character set choice	any
Ribbon color	any
Comment	(no default)
Alignment pattern	(no default)

To define the form, use one of the following commands

```
# lpforms -f form-name -F file-name ⮐
# lpforms -f form-name - ⮐
```

where *file–name* is the full path for the file.

The first command gets the form definition from a file; the second command gets the form definition from you, through the standard input. A *form–name* can be anything you choose, as long as it contains a maximum of fourteen alphanumeric characters and underscores.

If you need to change a form, just reenter one of the above commands. You need only provide information for items that must be changed; items for which you don't specify new information will stay the same.

Removing a Form

The LP print service imposes no fixed limit on the number of forms you may define. It is a good idea, however, to remove forms that are no longer appropriate. If you don't, users will see a long list of obsolete forms when choosing a form, and may be confused. In addition, because the LP print service must occasionally look through all the forms listed before performing certain tasks, the failure to remove obsolete forms may require extra, unnecessary processing by the print service.

To remove a form, enter the following command:

```
# lpforms -f form-name -x ⮐
```

Restricting User Access

If your system has a form that you don't want to make available to everyone, you can limit its availability to selected users. For example, you may want to limit access to checks to the people in the payroll department or accounts payable department.

The LP print service restricts the availability of a form by using the list of users allowed or denied access to that form. If a user is not allowed to use a particular form, the LP print service will reject his or her request to print a file with it.

The method used to allow or deny users access to a form is similar to the method used to allow or deny users access to the **cron** and **at** facilities. See the manual pages for **crontab**(1) and **cron**(1M). Briefly, the rules are as follows:

- An allow list is a list of users who are allowed to use the form. A deny list is a list of users who are not allowed to use the form.
- If the allow list is not empty, only the users listed are allowed; the deny list is ignored. If the allow list is empty, the users listed in the deny list are not allowed to to use the form. If both lists are empty, there are no restrictions on who may use the form.
- Specifying **all** in the allow list allows everybody to use the form; specifying **all** in the deny list allows no one except the user **lp** and the super-user (**root** or **sysadm**) to use the form.

If users on your system are to be able to access forms on a remote printer, it's necessary for all the users included on the allow list for the local system to be included on the allow list for the remote system, as well.

If, on the other hand, a local user is to be denied permission to use forms on a remote printer, it's not necessary for the deny lists on both the local and remote print services to include that user. By being included in only one of these deny lists, a user can be denied access to remote forms. As a courtesy to your users, however, it's a good idea to make sure that any local users who are included in a deny list on a remote system are included in the corresponding deny list on your local system. By doing this you can make sure that whenever a user on your system requests a form that he or she is not authorized to use, he or she is immediately informed that permission to use the form is being denied. If the local print service does not "know" that a user is denied permission to use a particular remote form, there will be a delay before the user receives a "permission denied" message from the remote system.

You can add names of users to either list using one of the following commands:

```
# lpforms -f form-name -u allow:user-list ↵
# lpforms -f form-name -u deny:user-list ↵
```

The *user-list* is a comma or space separated list of names of users. If you use spaces to separate the names, enclose the entire list (including the **allow:** or **deny:** but not the **-u**) in quotes. Each item in the list can include a system name, as shown under "User Access Restrictions" earlier in this section. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Specifying **allow:all** will allow everybody; specifying **deny:all** will deny everybody.

If you do not add usernames to the allow or deny lists, the LP print service will assume that everybody may use the form.

Alerting to Mount a Form

If you define more forms than printers, you will obviously not be able to print files on all the forms simultaneously. This means that some print requests may be held in a queue until you mount the forms they need. How will you know when to mount a particular form? One method would be to periodically monitor the number of print requests pending for that form. The LP print service, however, provides an easier way: You can ask to be alerted when the number of requests waiting for a form has exceeded a specified threshold.

You can choose one of several ways to receive an alert.

- You can receive an alert via electronic mail. See the manual page for the **mailx**(1) command.
- You can receive an alert written to any terminal on which you are logged in. See the manual page for the **write** command.
- You can receive an alert through a program of your choice.
- You can receive no alerts.

If you elect to receive no alerts, you are responsible for checking to see if any print requests haven't printed because the proper form isn't mounted. In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the form is mounted. You can choose the rate of repeated alerts, or choose to receive only one alert for each form.

To arrange for alerting to the need to mount a form, enter one of the following commands:

```
# lpforms -f form-name -A mail -Q requests -W minutes ↵
# lpforms -f form-name -A write -Q requests -W minutes ↵
# lpforms -f form-name -A 'command' -Q requests -W minutes ↵
```

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run the *command* for each alert. The shell environment in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and the current directory.

In each command line, the argument *requests* is the number of requests that need to be waiting for the form to trigger the alert, and the argument *minutes* is the number of minutes between repeated alerts. If you want mail sent or a message written to another user when a printer fault occurs, use the third command with the option **–A'mail** *username*' or **–A'write** *username*' .

If you want the print service to issue no alert when the form needs to be mounted, execute the following command:

```
# lpforms -f form-name -A none ↵
```

When you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts (for the current case only) by issuing the following command:

```
# lpforms -f form-name -A quiet ↵
```

Once the form has been mounted and unmounted again, alerts will resume if too many requests are waiting. Alerts will also start again if the number of requests waiting falls below the **–Q** threshold and then rises up to the **–Q** threshold again. This happens when waiting requests are canceled, and when the type of alerting is changed.

If *form–name* is **all** in any of the commands above, the alerting condition applies to all forms for which an alert has not already been defined.

If you don't define an alert method for a form, you will not receive an alert to mount it. If you define a method without the **-W** option, you will be alerted once for each occasion.

Mounting a Form

See "Mounting a Form or Print Wheel" under "Configuring Your Printers" in this section.

Examining a Form

Once you've defined a form to the LP print service, you can examine it with one of two commands, depending on the type of information you want to check. The **lpforms** command displays the attributes of the form. (The display produced by **lpforms** can be used as input; you may want to save it in a file for future reference.) The **lpstat** command displays the current status of the form.

Enter one of the following commands to examine a defined form.

```
# lpforms -f form-name -l ↵
# lpforms -f form-name -l > file-name ↵
# lpstat -f form-name ↵
# lpstat -f form-name -l ↵
```

The first two commands present the definition of the form; the second command captures this definition in a file, which can be used later to redefine the form if you inadvertently remove the form from the LP print service. The last two commands present the status of the form, with the second of the two giving a long form of output, similar to the output of **lpforms -l**:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set[,mandatory]
Ribbon color: ribbon-color
Comment:
comment
Alignment pattern: [ content-type]
content
```

To protect potentially sensitive content, the alignment pattern is not shown if the **lpstat** command is used.

Providing Filters

This section explains how you can manage the use of filters with the LP print service. You will see how you can

- define a new filter
- change a filter

- remove a filter
- examine a filter

The "Customizing the Print Service" section at the end of this section describes how you can write a filter. First, let's see what a filter is and how the LP print service can use one.

What is a Filter?

A filter is a program that you can use for any of three purposes:

- To convert a user's file from one data format to another so that it can be printed properly on a given printer
- To handle the special modes of printing that users may request with the **–y** option to the **lp** command (such as two-sided printing, landscape printing, draft or letter quality printing)
- To detect printer faults and notify the LP print service of them, so that the print service can alert you

Not every filter can perform all three tasks. Given the printer-specific nature of these three roles, the LP print service has been designed so that these roles can be implemented separately. This separation allows you or a printer manufacturer (or another source) to provide filters without having to change the LP print service.

A default filter is provided with the LP print service to provide simple printer fault detection; it does not convert files or handle any of the special modes. It may, however, be adequate for your needs.

Let's examine the three tasks performed by filters more closely.

Task 1: Converting Files

For each printer (local or remote) you can specify what file content types it can print. When a user submits a file to print on any printer, and specifies its content type, the print service will find a printer that can handle files of that content type. Because many applications can generate files for various printers, this is often sufficient. However, some applications may not generate files that can be printed on your printers.

By defining and creating a filter that converts such files into a type that your printers can handle, you can begin to support more applications in the LP print service. (The LP print service comes with a few filters for converting various types of files into PostScript.) For each filter you add to the system, you must specify one or more types of input it can accept and the type of output it can produce (usually only one).

When a user specifies (by executing **lp –T**) a file content type that no printer can handle, the print service tries to find a filter that can convert the file into an acceptable type. If the file to be printed is passed through a filter, the print service will then match the output type of that filter with a printer type or the input type of another filter. The LP print service will continue to match output types to input types in this way, thus passing a file through a series of filters, until the file reaches a printer that accepts it.

Below are some examples.

Example 1
The user Chris has run a spreadsheet program and has generated a file containing a copy of a spreadsheet. Chris now wants to print this file using the LP print service. You have only HP LaserJet printers on your system. Fortunately, the spreadsheet application understands how to generate output for several printers, and Chris knows it's necessary to request output that can be handled by the HP LaserJet. When Chris submits the file for printing, the LP print service queues it for one of the printers; no filter is needed.

Example 2
Marty has created a graphic image that can be displayed on a Tektronix 4014 terminal. Marty now wants to print this image, but all of the printers are PostScript printers. Fortunately, your system provides a filter called **posttek** that converts **Tektronix** type files to **PostScript**. Because you set the printer type to **PostScript**, the LP print service recognizes that it can use the **posttek** filter to convert Marty's output before printing it.

Task 2: Handling Special Modes

Another important role that filters can perform is the handling of special printing modes. Each filter you add to the filter table can be registered to handle special modes and other aspects of printing:

Special modes
Printer type
Character pitch
Line pitch
Page length
Page width
Pages to print
Character set
Form name
Number of copies

A filter is required to handle the special modes and printing of specific pages; the LP print service provides a default handling for all the rest. However, it may be more efficient to have a filter handle some of the rest, or it may be that a filter has to know several of these aspects to fulfill its other roles properly. A filter may need to know, for example, the page size and the print spacing if it is going to break up the pages in a file to fit on printed pages. As another example, some printers can handle multiple copies more efficiently than the LP print service, so a filter that can control the printer can use the information about the number of copies to skip the LP print service's default handling of multiple copies.

We'll see below how you can register special printing modes and other aspects of printing with each filter.

Task 3: Detecting Printer Faults

Just as converting a file and handling special printing modes is a printer-specific role, so is the detecting of printer faults. The LP print service attempts to detect faults in general, and for most printers it can do so properly. The range of faults that the print service can detect by itself, however, is limited. It can check for hang-ups (loss of carrier, the signal that indicates the printer is on-line) and excessive delays in printing (receipt of an XOFF flow-control character to shut off the data flow, with no matching XON to turn the flow back on). However, the print service can't determine the cause of a fault, so it can't tell you what to look for.

A properly designed filter can provide better fault coverage. Some printers are able to send a message to the host describing the reason for a fault. Others indicate a fault by using signals other than the dropping of a carrier or the shutting off of data flow. A filter can serve you by detecting more faults and providing more information about them than you would otherwise receive.

Another service a filter can provide is to wait for a printer fault to clear and then to resume printing. This service allows for more efficient printing when a fault occurs because the print request that was interrupted does not have to be reprinted in its entirety. Only a real filter, which has knowledge of the control sequences used by a printer, can "know" where a file breaks into pages; thus only such a filter can find the place in the file where printing should resume.

The LP print service has a simple interface that allows a filter to send you fault information and to restart printing if it can. The alerting mechanism (see "Printer Fault Alerting" under "Configuring Your Printers" in this section) is handled by the LP print service; the interface program that manages the filter takes all error messages from the filter and places them in an alert message that can be sent to you. Thus you'll see any fault descriptions generated by the filter. If you've set the printer configuration so that printing should automatically resume after a fault is cleared, the interface program will keep the filter active, so that printing can pick up where it left off.

Will Any Program Make a Good Filter?

It is tempting to use a program such as **troff**, **nroff**, or a similar word-processing program as a filter. However, the **troff** and **nroff** programs have a feature that allows references to be made in a source file to other files, known as *include files*. The LP print service does not recognize include files; it will not queue any that are referenced by a source file when that file is in a queue to be printed. As a result, the **troff** or **nroff** program, unable to access the include files, may fail. Other programs may have similar features that limit their use as filters.

Here are a few guidelines for evaluating a program for use as a filter:

- Only programs capable of reading data from standard input and writing data to standard output may be used as filters.

- Examine the kinds of files users will submit for printing that will require processing by the program. If they stand alone (that is, if they do not reference other files that the program will need), the program is probably okay.

  Check also to see if the program expects any files other than those submitted by a user for printing. If it does, those files must be in the directory of the person using the filter, and they must be readable by all users authorized to use the filter. The latter prerequisite is necessary because filters are run with the user ID and group ID of the user who submitted the print request.

- If referenced files are permitted in the files submitted for printing, or if the program will need files other than those submitted by a user, then the program, unable to access the additional files, is likely to fail. We suggest you don't use the program under consideration as a filter; instead, have users run the program before submitting files for printing.

Referenced files that are always specified by full pathnames *may* be okay, but only if the filter is used for local print requests. When used on requests submitted from a remote system for printing on your system, the filter may still fail if the referenced files exist only on the remote system.

Filters for Your System

The LP print service is delivered with several filters. As you add, change, or delete filters, you may overwrite or remove some of these original filters. If necessary, you can restore the original set of filters (and remove any filters you have added), with the following command:

```
# lpfilter -f all -i  ⤶
```

Defining a Filter

When adding a new filter, the first thing you must do is to define the characteristics of its use. To do this, issue the **lpfilter** command with arguments that specify the values of the following filter characteristics:

- the name of the filter (that is, a command name)
- the types of input it will accept
- the types of output it will produce
- the types of printers to which it will be able to send jobs
- the names of specific printers to which it will send jobs
- the type of the filter (whether it's a **fast** filter or a **slow** filter)
- options

Each of these characteristics is described below.

Command:
This is the full path of the filter program. If there are any fixed options that the program always needs, include them here.

Input types:
This is the list of file content types that the filter can process. The LP print service doesn't impose a limit on the number of input types that can be accepted by a filter, but most filters can take only one. Several file types may be similar enough so that the filter can deal with them. You can use whatever names you like here, using a maximum of fourteen alphanumeric characters and dashes (not underscores). Because the LP print service uses these names to match a filter with a file type, you should follow a consistent naming convention. For example, if more than one filter can accept the same input type, use the same name for that input type when you specify it for each filter. These names should be advertised to your users so they know how to identify the type of a file when submitting that file for printing.

Output types:
This is the list of file types that the filter can produce as output. For each input type the

filter will produce a single output type, of course; the output type may vary, however, from job to job. The names of the output types are also restricted to fourteen alphanumeric characters and dashes.

These names should either match the types of printers you have on your system, or match the input types handled by other filters. The LP print service groups filters together in a shell pipeline if it finds that several passes by different filters are needed to convert a file. It's unlikely that you will need this level of sophistication, but the LP print service allows it. Try to find a set of filters that takes (as input types) all the different files your users may want printed, and converts those files directly into types your printers can handle.

Printer types:

This is a list of printer types into which the filter can convert files. For most filters this list will be identical to the list of output types, but it can be different.

For example, you may have a printer that is given a single type for purposes of initialization (see "Printer Type" under "Configuring Your Printers" in this section), but which can recognize several different types of files. In essence this printer has an internal filter that converts the various types into one with which it can deal. Thus, a filter may produce one of several output types that match the "file types" that the printer can handle. The filter should be marked as working with that printer type.

As another example, you may have two different models of printers that are listed as accepting the same types of files. However, due to slight differences in manufacture, one printer deviates in the results it produces. You label the printers as being of different printer types, say A and B, where B is the one that deviates. You create a filter that adjusts files to account for the deviation produced by printers of type B. Because this filter is needed only for those printer types, you would list it as working only on type B printers.

For most printers and filters you can leave this part of the filter definition blank.

Printers:

You may have some printers that, although they're of the correct type for a filter, are in other ways not adequate for the output that the filter will produce. For instance, you may want to dedicate one printer for fast turn-around; only files that the printer can handle without filtering will be sent to that printer. Other printers, of identical type, you allow to be used for files that may need extensive filtering before they can be printed. In this case, you would label the filter as working with only the latter group of printers.

In most cases a filter should be able to work with all printers that accept its output, so you can usually skip this part of the filter definition.

Filter type:

The LP print service recognizes **fast** filters and **slow** filters. Fast filters are labeled **fast** because they incur little overhead in preparing a file for printing, and because they must have access to the printer when they run. A filter that is to detect printer faults has to be a fast filter.

Slow filters are filters that incur a lot of overhead in preparing a file and that don't require access to a printer. The LP print service runs slow filters in the background, without tying up a printer. This allows files that don't need slow filtering to move ahead; printers will not be left idle while a slow filter works on a file if other files can be printed simultaneously.

Slow filters that are invoked by modes (via the **–y** option), must be run on the computer where the print request was issued. The LP print service can't pass values for modes to remote systems. It can, however, match a file content type (specified after the **–T** option of the **lp** command) to a content type on a remote system. Therefore, if you want to activate special modes on a remote system, you must do so by specifying content types that will allow the LP print service to match input types and output types.

Options:

Options specify how different types of information should be transformed into command line arguments to the filter command. This information may include specifications from a user (with the print request), the printer definition, and the specifications implemented by any filters used to process the request.

There are thirteen sources of information, each of which is represented by a *keyword*. Each option is defined in a *template*, a statement in the following format: *keyword pattern=replacement*. This type of statement is interpreted by the **lpfilter** command to mean "When the information referred to by *keyword* has the value matched by *pattern*, take the *replacement* string, expand any regular expressions it contains, and append the result to the command line."

The options specified in a filter definition may include none, all, or any subset of these thirteen keywords. In addition, a single keyword may be defined more than once, if multiple definitions are required for a complete filter definition. (See "Defining Options with Templates" below.)

When you've gathered enough information to define the above characteristics of your filter, you are ready to run the **lpfilter** command, using your data as arguments. Because there are so many arguments, and because some of them may need to be entered more than once (with different values), we recommend you record this information first in a separate file and edit it, if necessary. You can then use the file as input to the **lpfilter** command and avoid typing each piece of information separately.

Whether you store the information in a file or enter it directly on the command line, use the following format:

```
Command: command-pathname [options]
Input types: input-type-list
Output types: output-type-list
Printer types: printer-type-list
Printers: printer-list
Filter type: fast or slow
Options: template-list
```

The information can appear in any order. Not all the information has to be given. When you do not specify values for the items appearing in Table 11–3, the values shown beside them are assigned by default.

Table 11–3 Filter Defaults

Item	Default
Command:	(no default)
Input types:	`any`
Output types:	`any`
Printer types:	`any`
Printers:	`any`
Filter type:	`slow`
Options:	(no default)

As you can see, the default values define a very flexible filter, so you probably have to supply at least the input and output types. When you enter a list, you can separate the items in it with blanks or commas, unless it is a *templates–list*; items in a *templates–list* must be separated by commas.

Defining Options with Templates

A template is a statement in a filter definition that defines an option to be passed to the filter command based on the value of one of the characteristics of the filter. A filter definition may include more than one template. Multiple templates may be entered on a single line and separated with commas, or they may be entered on separate lines, preceded by the **Options:** prefix.

The format of a template is as follows:

```
keyword pattern = replacement
```

The *keyword* identifies the type of option being registered for a particular characteristic of the filter.

Let's look at an example of how an option is defined for a particular filter. Suppose you want to have the print service scheduler assign print requests to filters on the basis of the following criteria:

- If the type of output to be produced by the filter is **impress**, then pass the **–I** option to the filter.
- If the type of output to be produced by the filter is **postscript**, then pass the **–P** option to the filter.

To specify these criteria, provide the following templates as options to the lpfilter command.

```
Options: OUTPUT impress=–I, OUTPUT postscript=–P
```

If the **Options:** line becomes too long, put each template on a separate line, as follows:

```
Options: OUTPUT impress=–I
Options: OUTPUT postscript=–P
```

In both templates, the *keyword* is defined as **OUTPUT**. In the first template, the value of *pattern* is **impress** and the value of the *replacement* is **–I**. In the second template, the value of *pattern* is **postscript** and the value of the *replacement* is **–P**.

Template Keywords

Table 11–4 shows the thirteen keywords available for defining options in a filter definition.

Table 11–4 Filter Option Keywords

Characteristic	Keyword	Possible Patterns	Example
Content type (input)	INPUT	content–type	troff
Content type (output)	OUTPUT	content–type	postscript
Printer type	TERM	printer–type	att495
Printer name	PRINTER	printer–name	lp1
Character pitch	CPI	scaled–decimal	10
Line pitch	LPI	scaled–decimal	6
Page length	LENGTH	scaled–decimal	66
Page width	WIDTH	scaled–decimal	80
Pages to print	PAGES	page–list	1–5,13–20
Character set	CHARSET	character–set	finnish
Form name	FORM	form–name	invoice2
Number of copies	COPIES	integer	3
Special modes	MODES	mode	landscape

To find out which values to supply for each type of template (that is, for the *pattern* and *replacement* arguments for each *keyword*), see the source of information listed below.

- The values for the **INPUT** and **OUTPUT** templates come from the file type that needs to be converted by the filter and the output type that has to be produced by the filter, respectively. They'll each be a type registered with the filter.
- The value for the **TERM** template is the printer type.
- The value for the **PRINTER** template is the name of the printer that will be used to print the final output.
- The values for the **CPI**, **LPI**, **LENGTH**, and **WIDTH** templates come from the user's request, the form being used, or the default values for the printer.
- The value for the **PAGES** template is a list of pages that should be printed. Typically, it is a comma separated list of page ranges, each of which consists of a dash separated pair of numbers or a single number (such as **1–5,6,8,10** for pages 1 through 5, 6, 8, and 10). However, whatever value was given in the **–P** option to a print request is passed unchanged.

- The value for the **CHARSET** template is the name of the character set to be used.
- The value for the **FORM** template is the name of the form requested by the **–f** option of the **lp** command.
- The value of the **COPIES** template is the number of copies that should be made of the file. If the filter uses this template, the LP print service will reduce to 1 the number of copies of the filtered file it will print, since this single copy will actually comprise the multiple copies produced by the filter.
- The value of the **MODES** template comes from the **–y** option of the **lp** command (the command used to submit a print request). Because a user can specify several **–y** options, there may be several values for the **MODES** template. The values will be applied in the left-to-right order given by the user.

The *replacement* part of a template shows how the value of a template should be given to the filter program. It is typically a literal option, sometimes with the place-holder * included to show where the value goes. The *pattern* and *replacement* can also use the regular expression syntax of **ed**(1) for more complex conversion of user input options into filter options. All of the regular expression syntax of **ed**(1) is supported, including the **\(** ... **\)** and **\n** constructions, which can be used to extract portions of the *pattern* for copying into the *replacement*, and the **&**, which can be used to copy the entire *pattern* into the *replacement*. If a comma or an equals sign (=) is included in a *pattern* or a *replacement*, escape its special meaning by preceding it with a backslash (\). Note that some regular expressions include commas that will have to be escaped this way. A backslash in front of any of these characters is removed when the *pattern* or *replacement* is used.

The following examples show how this works.

Example 1
You provide the following filter definition for a filter called **col**.

```
Input types:  N37, Nlp, simple
Output types:  simple
Command:  /usr/bin/col
Options:  TERM 450 = –b, MODES expand = –x
Options:  INPUT simple = –p –f
```

If you provide more than one definition (that is, more than one line) for any filter characteristic other than **OPTIONS**, only the second definition will be used by the print service.

After you have registered this definition with the print service by entering it as input with the **lpfilter** command, users' print requests will be handled as follows:

If a user enters the command

```
# lp –y expand report.dec10 ↵
```

the filter command will be run with the following arguments:

```
/usr/bin/col -x -p -f
```

If a user enters the command

```
# lp -T N37 -y expand report.dec10 ↵
```

the filter command will be run with the following arguments:

```
/usr/bin/col -x
```

Qualifier: The default printer is not of type **450**.

If a user enters the command

```
# lp -y expand -T 450 report.dec10 ↵
```

the filter command will be run with the following arguments:

```
/usr/bin/col -b -x
```

Example 2
The filter program is called **/usr/lib/lp/postscript/dpost**. It takes one input type, **troff**, produces an output type called **postscript**, and works with any printer of type **postscript**. You've decided that your users need give just the abbreviations **port** and **land** when they ask for the paper orientation to be portrait mode and landscape mode, respectively. Because these options are not intrinsic to the LP print service, users must specify them using the **-y** option to the **lp** command.

The filter definition would look like this:

```
Input types: troff
Output types: postscript
Printer types: postscript
Filter type: slow
Command: /usr/lib/lp/postscript/dpost
Options: LENGTH * = -l *, CHARSET * = -s *
Options: MODES port = -o portrait, MODES land = -o landscape
```

A user submitting a file of type **troff** for printing on a PostScript printer (type **postscript**), with requests for landscape orientation and the **gothic** character set, would enter the following command:

```
# lp -T troff -S gothic -y land -d any ↵
```

Then this filter would be invoked by the LP print service to convert the file as follows:

```
/usr/lib/lp/postscript/dpost -S gothic -o landscape
```

Example 3
You add the following option template to the previous example:

```
Options:  MODES size\=\([0-9]*\)x\([0-9]*\) = -h\1 -w\2
```

This template is used to convert a **MODES** option of the form

```
-y size=heightxwidth
```

into a pair of filter options,

```
-hheight -wwidth
```

So if a user gives the following command

```
# lp -y size=24x80 ↵
```

the **dpost** command would include the following options:

```
-h24 -w80
```

Command to Enter

Once a filter definition is complete, enter one of the following commands to add the filter to the system.

```
# lpfilter -f filter-name -F file-name ↵
# lpfilter -f filter-name - ↵
```

The first command gets the filter definition from a file, and the second command gets the filter definition from the standard input. A *filter–name* can be any string you choose, with a maximum of fourteen alphanumeric characters and underscores.

If you need to change a filter, just reenter one of the same commands. You need only provide information for those items that must be changed; items for which you don't specify new information will stay the same.

Removing a Filter

The LP print service imposes no fixed limit on the number of filters you can define. It is a good idea, however, to remove filters no longer applicable, to avoid extra processing by the LP print service which must examine all filters to find one that works in a given situation.

To remove a filter, enter the following command:

```
# lpfilter -f filter-name -x ↵
```

Examining a Filter

Once you've added a filter definition to the LP print service, you can examine it by running the **lpfilter** command. The output of this command is the filter definition displayed in a format that makes it suitable as input. You may want to save this output in a file that you can use later to redefine the filter if you inadvertently remove the filter from the LP print service.

To examine a defined filter, enter one of the following commands:

```
# lpfilter -f filter-name -l ↵
# lpfilter -f filter-name -l >file-name ↵
```

The first command presents the definition of the filter on your screen; the second command captures this definition in a file for future reference.

Restoring Factory Defaults

The software is shipped from the factory with default values set for all options. If, after changing these defaults, you want to reset them, execute the following command:

```
# lpfilter -f filter-name -i ↵
```

You can restore an individual filter by giving its name in place of *filter–name*, or you can restore everything by giving the name **all**.

A Word of Caution

Adding, changing, or deleting filters can cause print requests still queued to be canceled. This is because the LP print service evaluates all print requests still queued, to see which are affected by the filter change. Requests that are no longer printable, because a filter has been removed or changed, are canceled (with notifications sent to the people who submitted them). There can also be delays in the responses to new or changed print requests when filters are changed, due to the many characteristics that must be evaluated for each print request still queued. These delays can become noticeable if there is a large number of requests that need to be filtered.

Because of this possible impact, you may want to make changes to filters during periods when the LP print service is not being used much.

Managing the Printing Load

Occasionally you may need to stop accepting print requests for a printer or move pending print requests from one printer to another. There are various reasons why you might want to do this, such as the following:

- the printer needs periodic maintenance
- the printer is broken
- the printer has been removed
- you've changed the configuration so that the printer is to be used differently
- too many large print requests are queued for one printer and should be spread around

If you are going to make a big change in the way a printer is to be used, such as stopping its ability to handle a certain form, changing the print wheels available for it, or disallowing some users from using it, print requests that are currently queued for printing on it will have to be moved or canceled. The LP print service will attempt to find alternate printers, but only if the user doesn't care which printer is to be used. Requests for a specific printer won't be automatically moved; if you don't move them first, the LP print service will cancel them.

If you decide to take a printer out of service, to change its configuration, or to lighten its load, you may want to move print requests off it and reject additional requests for it for awhile. To do so, use the **lpmove** and **reject** commands. If you do reject requests for a printer, you can accept requests for it later, by using the **accept** command.

Rejecting Requests for a Printer or Class

To stop accepting any new requests for a printer or class of printers, enter the following command.

```
# reject -r "reason"  printer-or-class-name ↵
```

You can reject requests for several printers or classes in one command by listing their names on the same line, separating the names with spaces. The *"reason"* will be displayed whenever anyone tries to print a file on the printer. You can omit it (and the **-r**) if you don't want to specify a reason.

Although the **reject** command stops any new print requests from being accepted, it will not move or cancel any requests currently queued for the printer. These will continue to be printed as long as the printer is enabled.

Accepting Requests for a Printer or Class

After the condition that led to rejecting requests has been corrected or changed, enter one of the the following commands to start accepting new requests.

```
# accept printer-name ↵
# accept class-name ↵
```

Again, you can accept requests for several printers or classes in one command by listing their names on the same line.

You will always have to use the **accept** command for a new printer or class after you have added it, because the LP print service does not initially accept requests for new printers or classes.

Moving Requests to Another Printer

If you specify **-d any** when you run the **lp** command to queue a job, the print service schedules the job for a particular printer. If another becomes available first, the job is sent to the latter printer. If a job is scheduled for a given printer and you run **lpmove** to get jobs off that printer, that job will be moved off and the destination will change from **any** to the printer you've specified on the **lpmove** command line. Users may not have intended this side effect. If not, run the following command:

```
# lp -i request-ID -d any  ↵
```

This command will change the destination for the requested job to the original destination: **any** (that is, any available printer).

If you have to move requests from one printer or class to another, enter one of the following commands

```
# lpmove request-id printer-name
# lpmove printer-name1 printer-name2
```

You can give more than one request ID before the printer name in the first command.

The first command above moves the listed requests to the printer *printer–name*. The second command tries to move *all* requests currently queued for *printer–name1* to *printer–name2*. If some requests cannot be printed on the new printer, they will be left in the queue for the original printer. When the second command is used, the LP print service also stops accepting requests for *printer–name1* (the same result you would obtain by running the command **reject** *printer–name2*).

Examples

Here are some examples of how you might use these three commands.

Example 1

You've decided it is time to change the ribbon and perform some preventive maintenance on printer **lp1**. First, to prevent the loss of print requests already queued for **lp1**, you move all requests from printer **lp1** to printer **lp2**.

```
# lpmove lp1 lp2
```

After the requests are moved, make sure the LP print service does not print any more requests on **lp1** by disabling it.

```
# disable lp1
```

Now you may physically disable the printer and start working on it.

Example 2

You've finished changing the ribbon and doing the other work on **lp1**; now it's time to bring it back into service. Execute the following commands in any order:

```
# accept lp1
# enable lp1
```

See "Enabling and Disabling a Printer" under "Making Printers Available" in this section.

Example 3

You notice that someone has queued several large files for printing on the printer **laser1**. Meanwhile **laser2** is idle because no one has queued requests for it. Move the two biggest requests (**laser1–23** and **laser1–46**) to **laser2**, and reject any new requests for **laser1** for the time being.

```
# lpmove  laser1–23  laser1–46  laser2 ⤶
# reject  –r"too busy––will reopen later"  laser1 ⤶
```

Managing Queue Priorities

The LP print service provides a simple priority mechanism that users can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the user who submits it; this is a number from 0 to 39, with *smaller* numbers indicating *higher* levels of priority. Requests with higher priority (smaller numbers) are placed ahead of requests with lower priority (larger numbers).

Thus, for example, a user who decides that her print request is of low priority can assign it a larger value when she submits the file for printing. Another user who decides that his print request is of high priority can assign it a smaller value when he submits the file for printing.

A priority scheme this simple would not work if there were no controls on how high one can set the priority. You can define the following characteristics in this scheme:

- Each user can be assigned a priority limit. One cannot submit a print request with a priority higher than his or her limit, although one can submit a request with a lower priority.
- A default priority limit can be assigned for the balance of users not assigned a personal limit.
- A default priority can be set. This is the priority given print requests to which the user does not assign a priority.

By setting the characteristics according to your needs, you can prevent lower priority printing tasks (such as regular printing by most staff members) from interfering with higher priority printing tasks (such as payroll check printing by the accounting staff).

You may find that you want a critical print request to print ahead of any others, perhaps even if it has to preempt the currently printing request. You can have the LP print service give immediate handling to a print request, and you can have it hold another print request. This will allow the first request to be printed and will delay the second print request until you allow it to be resumed.

The **lpusers** command lets you assign both priority limits for users and priority defaults. In addition, you can use the **lp –i***request–id* **–Hhold** and **lp –i***request–id* **–Himmediate** commands to put a request on hold or to move it up for immediate printing, respectively. These commands are discussed in detail below.

Setting Priority Limits

To set a user's priority limit, enter the following command.

```
# lpusers –q priority-level –u username ⤶
```

You can set the limit for a group of users by listing their names after the **–u** option. Separate multiple names with a comma or space (enclose the list in quotes if you use a space, though). The argument *priority–level* is a number from 0 to 39. As mentioned before, the lower the number the higher the priority, or, in this case, the priority limit.

If you want to set a priority limit for the remaining users, enter the following command:

```
# lpusers –q priority–level ↵
```

This sets the default limit; the default applies to those users for whom you have not set a personal limit, using the first **lpusers** command.

If you later decide that someone should have a different priority limit, just reenter the first command above with a new limit. Or, if you decide that the default limit is more appropriate for someone who already has a personal limit, enter the following command:

```
# lpusers –u username ↵
```

Again, you can do this for more than one user at a time by including a list of names. Using the **lpusers** command with just the **–u** option removes users' personal priority limits and puts the default limit into effect for those users.

Setting a Default Priority

To set the default priority (the priority level assigned to print requests submitted without a priority), use the following command:

```
# lpusers –d priority–level ↵
```

Don't confuse this default with the default limit. This default is applied when a user doesn't specify a priority level; the default limit is applied if you haven't assigned a limit for a user—it is used to limit the user from requesting too high a priority. If the default priority is greater than the limit for a user, the limit is used instead.

If you do not set a default priority, the LP print service will use a default of 20.

Examining the Priority Limits and Defaults

You can examine all the settings you have assigned for priority limits and defaults by entering the following command.

```
# lpusers –l ↵
```

Moving a Request Around in the Queue

Once a user has submitted a print request, you can move it around in the queue to some degree:

- you can adjust the priority to any level, regardless of the limit for the user (who may adjust it only up to his or her limit)

- both you and the user can put it on hold and allow other requests to be printed ahead of it
- you can put it at the head of the queue for immediate printing

Use the **lp**(1) command to do any of these tasks.

Changing the Priority for a Request

If you want to change the priority of a particular request that is still waiting to be printed, you can assign a new priority level to it. By doing so, you can move it in the queue so that it is ahead of lower priority requests, and behind requests at the same level or of higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because, as the administrator, you can override this limit.

Enter the following command to change the priority of a request.

```
# lp –i request-ID –q new-priority-level ⤶
```

You can change only one request at a time with this command.

Putting a Request on Hold

Any request that has not finished printing can be put on hold. This will stop its printing, if it is currently printing, and keep it from printing until you resume it. A user may also put his or her own request on hold and then resume it, but may not resume a print request that has been put on hold by the administrator.

To place a request on hold, enter the following command:

```
# lp –i request-ID –H hold ⤶
```

Enter the following command to resume the request:

```
# lp –i request-ID –H resume  ⤶
```

Once resumed a request will continue to move up the queue and will eventually be printed. If printing had already begun when you put it on hold, it will be the next request printed.

Moving a Request to the Head of the Queue

You can move a print request to the head of the queue where it will be the next one eligible for printing. If you want it to start printing immediately but another request is currently being printed, you may interrupt the first request by putting it on hold, as described above.

Enter the following command to move a print request to the head of the queue:

```
# lp –i request-ID –H immediate ⤶
```

Only you, as the administrator, can move a request in this way; regular users cannot use the **–Himmediate** option. If you set more than one request for immediate printing, the requests will be printed in the reverse order set; that is, the request moved to the head of the queue most recently will be printed first.

Starting and Stopping the LP Print Service

Under normal operation, you should never have to start or stop the LP print service manually. The system starts it each time the system boots, and stops it each time the system stops. If, however, you need to stop the LP print service without stopping the entire system, do so by following the procedure described below.

Stopping the LP print service will cause all printing to cease within seconds. Any print requests that have not finished printing will be printed in their entirety after the LP print service is restarted. The printer configurations, forms, and filters in effect when the LP print service is stopped will be restored after it is restarted. To start and stop the LP print service manually, you must be logged in as either the user **lp** or the super-user (**root** or **sysadm**).

Manually Stopping the Print Service

To stop the LP print service manually, enter the following command:

```
# lpshut ↵
```

The message

```
Print services stopped.
```

will appear, and all printing will cease within a few seconds. If you try to stop the LP print service when it is not running, you will see the message

```
Print services already stopped.
```

Manually Starting the Print Service

To restart the LP print service manually, enter the following command:

```
# lpsched ↵
```

The message:

```
Print services started.
```

will appear. It may take a minute or two for the printer configurations, forms, and filters to be reestablished, before any saved print requests start printing. If you try to restart the LP print service when it is already running, you will see the message

```
Print services already active.
```

The LP print service does not have to be stopped to change printer configurations or to add forms or filters.

Managing the LP Print Service Logs

The LP print service has several logs that accumulate information on various LP services. By default, the LP print service does not remove or truncate these logs; therefore, you need to make sure they do not consume too much disk space. The easiest way to manage these logs is by submitting the **cron** jobs provided for the purpose in the LP print service prototype **crontab** file, **/admin/crontabs/lp.proto**. For more information on submitting **cron** jobs, see Chapter 2. Also see the manual pages for **cron**(1M) and **crontab**(1).

Located in **/var/spool/lp/logs**, the LP print service logs are **lpNet**, **lpsched**, and **requests**. The following sections elaborate.

The lpNet Log

The LP system runs a process called **lpNet**, which handles communication with LP systems running on other systems in your network. The **lpNet** process logs its activities to the file **/var/lp/logs/lpNet**.

As the **lpNet** file grows, you should clean it out occasionally. The easiest way to do this is with the appropriate **cron** job from **/admin/crontabs/lp.proto**. The job for this purpose appears below.

```
17 3 * * 0 /usr/lbin/agefile -c4 /var/lp/logs/lpNet
```

This **cron** job ages the **lpNet** log so that every week starts a new **lpNet** log, and no log remains on the system for more than five weeks. This job runs every Sunday at 3:17 am.

The lpsched Log

The LP print service keeps a log for scheduler–related events. This log is **/var/spool/lp/logs/lpsched**.

As with the **lpNet** log, the best way to control the size of the **lpsched** log is by submitting the appropriate **cron** job from **/admin/crontabs/lp.proto**. The job for this purpose appears below.

```
15 3 * * 0 /usr/lbin/agefile -c4 /var/lp/logs/lpsched
```

This **cron** job ages the **lpsched** log so that every week starts a new **lpsched** log, and no log remains on the system for more than five weeks. This job runs every Sunday at 3:15 am.

The requests Log

The **/var/spool/lp/logs/requests** file accumulates information about each completed request. Initially, the directories **/var/spool/lp/tmp/***system* and **/var/spool/lp/requests/***system* contain files that describe each request that has been submitted to the LP print service. Each request has two files (one in each directory) that contain information about the request. The information is split to put more sensitive information in the **/var/spool/lp/requests/***system* directory where it can be kept secure: the request file in the **/var/spool/lp/tmp/***system* is safe from all except the user who submitted the request, while the file in **/var/spool/lp/requests/***system* is safe from all users, including the submitting user.

These files remain in their directories only as long as the request is in the queue. Once the request is finished, the information in the files is combined and appended to the file **/var/lp/logs/requests**.

Log Structure

The **requests** log has a simple structure that makes it easy to extract data from it using common shell commands. Requests are listed in the order they are printed, and are separated by lines showing their request IDs. Each line below the separator line is marked with a single letter that identifies the kind of information contained in that line. Each letter is separated from the data by a single space. See the following list for details.

=	This is the separator line. It contains the request ID, the user and group IDs of the user, the total number of bytes in the original (unfiltered) files, and the time when the request was queued. These items are separated by commas and are in the order just named. The user ID, group ID, and sizes are preceded by the words **uid**, **gid**, and **size**, respectively.
C	The number of copies printed.
D	The printer or class destination or the word **any**.
F	The name of the file printed. This line is repeated for each file printed; files were printed in the order given.
f	The name of the form used.
H	One of three types of special handling: **resume**, **hold**, and **immediate**. The only useful value found in this line will be **immediate**.
N	The type of alert used when the print request was successfully completed. The type is the letter **M** if the user was notified by mail, or **W** if the user was notified by a message to his or her terminal.
O	The **–o** options.
P	The priority of the print request.
p	The list of pages printed.
r	This single letter line is included if the user asked for *raw* processing of the files (the **–r** option of the **lp** command).
S	The character set or print wheel used.
s	The outcome of the request, shown as a combination of individual bits expressed in hexadecimal form. While several bits are used internally by the print service, the most important bits are listed below:

`0x0004`	Slow filtering finished successfully.
`0x0010`	Printing finished successfully.
`0x0040`	The request was canceled.
`0x0100`	The request failed filtering or printing.

T	The title placed on the banner page.
t	The type of content found in the files.

U	The name of the user who submitted the print request.
x	The slow filter used for the request.
Y	The list of special modes to give to the filters used to print the request.
y	The fast filter used for the request.
z	The printer used for the request. This will differ from the destination (the **D** line) if the request was queued for any printer or a class of printers, or if the request was moved to another destination by the LP print service administrator.

Cleaning out the requests Log

The LP print service does not remove the **requests** file; therefore, the file will grow indefinitely if you do not remove it or shorten it occasionally. The best way to do this is with the **cron** job provided in the LP print service's prototype **crontab** file, **/admin/crontabs/lp.proto**. The prototype file contains three **cron** jobs. The one for managing the **requests** file appears below.

```
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
```

This jobs appears as one line in the **crontab** file, but it is split into several lines here for readability.

What this entry does, briefly, is *age* the file, changing the name to **requests–1**, and moving the previous day's copy to **requests–2**. The number **2** in the **–c** option to the **agefile** program keeps the log files from the previous two days, discarding older log files. By changing this number you can change the amount of information saved. On the other hand, if you want the information to be saved more often, or if you want the file to be cleaned out more often than once a day, you can change the time when the **crontab** entry is run by changing the first two numbers. The current values, **13** and **3**, cause cleaning up to be done at 3:13am each day.

The default **crontab** entry supplied is sufficient to keep the old print request records from accumulating in the spooling file system. You may want to condense information in the request log to produce a report on the use of the LP print service, or to aid in generating accounting information. You can produce a different script that examines the file and extracts information just before the clean up procedure.

PostScript Printers

PostScript is a general purpose programming language, like C or Pascal. In addition to providing the usual features of a language, however, PostScript allows a programmer to specify the appearance of both text and graphics on a page.

A PostScript printer is a printer equipped with a computer that runs an interpreter for processing PostScript language files. When a PostScript printer receives a file, it runs that file through the interpreter and then prints it. Unless special provisions have been made by the manufacturer, files submitted to a PostScript printer must be written in the PostScript language.

Why would you want to use a PostScript printer? PostScript provides excellent facilities for managing text and graphics and combining them. Graphics operators facilitate the construction of geometric figures which can then be positioned and scaled with any orientation. The text capabilities allow the user to specify a number of different fonts that can be placed on a page in any position, size, or orientation. Because text is treated as graphics, text and graphics are readily combined. Moreover, the language is resolution and device independent, so that draft copies can be proofed on a low-resolution device and the final version printed in higher resolution on a different device.

Applications that support PostScript, including word-processing and publishing software, will create documents in the PostScript language without intervention by the user. Thus, it is not necessary to know the details of the language to take advantage of its features. However, standard files that many applications produce cannot be printed on a PostScript printer because they are not described in the language. The LP print service provides optional filters to convert many of these files to PostScript so that users may take advantage of PostScript and continue to use their standard applications such as **troff**.

How to Use a PostScript Printer

When the PostScript printers and filters have been installed, the LP subsystem manages PostScript files like any others. If **psfile** is a file containing a PostScript document and **psprinter** has been defined as a PostScript printer, the command

```
# lp -d psprinter -Tpostscript psfile ↵
```

will schedule the print request and manage the transmission of the request to the PostScript printer.

Support of Non-PostScript Print Requests

A PostScript printer may not be able to interpret every kind of file that an application sends to it. The following list names a few of the formats that your printer may not be able to accommodate.

troff	Print a **troff** text file.
simple	Print an ASCII text file.
dmd	Print the contents of a bit-mapped display (for example, an AT&T 630).
tek4014	Print files formatted for a Tektronix 4014 device.
daisy	Print files intended for a daisy-wheel printer (for example, a Diablo 630).
plot	Print plot-formatted files

Optional filters are provided with the LP print service to translate print requests with these formats to the PostScript language. For example, to convert a file containing ASCII or **troff** code to PostScript code, the filter takes that text and writes a program around it, specifying printing parameters such as fonts and the layout of the text on a page.

Once the PostScript filters are installed, they will be invoked automatically by the LP print service when a user specifies a content-type for a print request with the **–T** option. For example,

```
# lp -d psprinter -T simple report2 ↵
```

will automatically convert the ASCII file **report2** (a file with an ASCII or **simple** format) to PostScript when the destination printer *psprinter* has been defined to the system as a PostScript printer.

Additional PostScript Capabilities Provided by Filters

The filters previously described also take advantage of PostScript capabilities to provide additional printing flexibility. Most of these features may be accessed through the mode option (invoked by the **–y** option) to the **lp** command. These filters allow you to use several unusual options for your print jobs. The following list describes these options and shows the option you should include on the **lp** command line for each one.

–y reverse
Reverse the order in which pages are printed.

–y landscape
Change the orientation of a physical page from portrait to landscape.

–y x=x*number*,y=y*number*
Change the default position of a logical page on a physical page by moving the origin.

–y group=*number*
Group multiple logical pages on a single physical page.

–P *number*
Select, by page numbers, a subset of a document to be printed

If these filters are to be used with an application that creates PostScript output, make sure that the application conforms to the PostScript file structuring comments. In particular, the beginning of each PostScript page must be marked by the comment

```
%%Page:label ordinal
```

where `ordinal` is a positive integer that specifies the position of the page in the sequence of pages in the document.

For example, say you have a file called **report2** that has a content type **simple** (meaning that the content of this file is in ASCII format). You want to print six pages of this file (pages 4-9) in landscape mode (that is, sideways on the page) with two logical pages on each physical page. Because one of the printers on your system (**psprinter**) is a PostScript printer, you can do this, by entering the following command:

```
# lp –d psprinter –T simple –P 4–9 –y landscape,group=2 report2 ⤶
```

In addition, the LP print service offers a special filter that can print a gray-scale representation of a matrix. (A gray-scale matrix is a matrix in which each cell is colored one of seven shades of gray to indicate the value of the cell. Darker shades correspond to larger values.) To print a gray-scale representation, specify **matrix** as the content type of your source file by giving the **–T matrix** option.

The Administrator's Duties

Support of PostScript printers is similar to support of other printers, in that the printers must be defined to the system with the **lpadmin** command and the appropriate software must be installed to manage them. PostScript printers may require some additional effort in supporting fonts.

Installing and Maintaining PostScript Printers

PostScript printers, like other printers, are installed with the **lpadmin** command. The content type of a PostScript printer must be consistent with the content type used in PostScript filters. Consequently, you should install serial PostScript printers with a content type of **PS**, causing LP to use the **postio** filter to communicate with the printer. When installing a parallel printer, however, specify a content type of **postscript** instead, causing LP not to use the **postio** filter. The **postio** filter requires a bidirectional communication connection, which can only be supplied by serial lines.

The content type accepted by the printer is specified with the **–I** option in the **lpadmin** command. In addition, you must tell the LP print service which fonts are available on the printer. To do so, enter this information in the font list for each printer.

Installing and Maintaining PostScript Filters

PostScript filters are provided with DG/UX Release 5.4, but to use them, you need to add them to the LP system with the Add Filter operation. This section contains specific information about the location and function of these filters. In certain circumstances, you may find it helpful to change existing filter descriptions.

PostScript filters are contained in the directory **/usr/lib/lp/filter/postscript.**

There are two types of filters: fast filters and slow filters. For a definition of these types, see the **lpfilter**(1M) manual page and "Defining a Filter" earlier in this section.

A prerequisite of communication between any system and a serial PostScript printer is the presence of the **postio** filter on the system. This program is the only mandatory PostScript filter for serial PostScript printers. The following filters allow other types of documents to be translated to PostScript and to be printed on a PostScript printer.

File Content Type	Filter
simple	**postprint**
troff	**dpost**
daisy (as for Diablo 630)	**postdaisy**
dmd (as for AT&T 630)	**postdmd**
tek4014 (Tektronix)	**posttek**
plot	**postplot**

The following filters perform special functions:

Function	Filter
Communicate with printer	**postio**
Download fonts	**download**
Reverse or select pages	**postreverse**
Matrix gray scales	**postmd**

Installing and Maintaining PostScript Fonts

One of the advantages of PostScript is its ability to manage fonts. Fonts are stored in outline form, either on the printer or on a host that communicates with a printer. When a document is printed, the PostScript interpreter generates each character as needed (in the appropriate size) from the outline description of it. If a font required for a document is not stored on the printer being used, it must be transmitted to that printer before the document can be printed. This transmission process is called "downloading fonts."

Fonts are stored and accessed in several ways.

- Fonts may be stored permanently on a printer. These "printer resident" fonts may be installed in ROM on the printer by the manufacturer. If the printer has a disk, fonts may be installed on that disk by you (that is, by the print service administrator). Most PostScript printers are shipped with thirty-five standard fonts.
- A font may be "permanently-downloaded" by being transmitted to a printer with a PostScript "exitserver" program. A font downloaded in this way will remain in the printer's memory until the printer is turned off. Memory allocated to this font will reduce the memory available for PostScript print requests. Use of exitserver programs requires the printer system password and may be reserved for the printer administrator. This method is useful when there is continual use of a font by the majority of print requests serviced by that printer.
- Fonts may be prefixed to a user's print request by the user, and be transmitted as part of the user's print request. When the user's document has been printed, the space allocated to the font is freed for other print requests. The font is stored in the user's directory. This method is preferable for fonts with more limited usage.
- Fonts may be stored on a system shared by many users. These fonts may be described as "host-resident." This system may be a server for the printer or may be a system connected to the printer by a network. Each user may request fonts in the document to be printed. This method is useful when there are a large number of available fonts or there is not continual use of these fonts by all print requests. If the fonts will be used only on printers attached to a server, they should be stored on the server. If the fonts are to be used by users on one system, who may send jobs to multiple printers on a network, they may be stored on the users' system.

The LP print service allows you to manage fonts in any of three ways. It provides a special download filter to manage fonts using the last method described above.

Managing Printer-Resident Fonts

Most PostScript printers come equipped with fonts resident in the printer ROM. Some printers have a disk on which additional fonts are stored. When a printer is installed, the list of printer-resident fonts should be added to the font-list for that printer. These lists are kept in the printer administration directories. For a particular printer, this list is contained in the file

/etc/lp/printers/*printer-name***/residentfonts**

where *printer–name* is the name of the printer.

When fonts are permanently downloaded to the printer, the font names should be added to this file. If the printer is attached to a remote system, this list should include fonts which reside on that system and are available for downloading to the printer. This prevents fonts from being transmitted unnecessarily across a network. These files must be edited manually; that is, with the help of a text editor such as **vi**(1).

Installing and Maintaining Host-Resident Fonts

Some fonts will be resident on the host and transmitted to the printer as needed for particular print requests. As the administrator, it's your job to make PostScript fonts available to all the users on a system. To do so, you must know how and where to install these fonts, using the guidelines described previously.

Install host-resident PostScript fonts by copying the font files to the appropriate directory.

Where Are Fonts Stored?
The fonts available for use with PostScript printers reside in directories called **/usr/lib/font/dev***device*, where *device* is a name representing an output device. Each device directory contains two files for each font that it supports. One file, *font***.name**, contains the full name of the font. The second file, *font***.out**, contains data that the printer requires to print the font. The *font* part of the file names typically consists of two characters. The first character represents the typeface, such as **H** for Helvetica, **C** for Courier, and so on. The second character represents the particular font, such as **B** for bold, **I** for Italic, and so on. For example, the file **HB.name** contains the full name of the Helvetica Bold font:

```
% cat  HB.name ⤶
Helvetica-Bold
```

Downloading Host-Resident Fonts

The creators of the PostScript language anticipated that users would want to download fonts to printers. For this purpose, they defined a standard set of *structuring conventions* for PostScript programs. The download filter relies on these structuring conventions to determine which fonts must be downloaded. See your PostScript documentation.

When the LP print service receives a request for a job that requires fonts not loaded on the printer, it forwards that request to a filter that downloads fonts. The request to download is made through the **-y download** option. Alternatively, an administrator may reinstall this filter so that all PostScript documents would be examined to determine if fonts were needed. The **-y** option would then become unnecessary.

The download filter does five things:

- It searches the PostScript document to determine which fonts have been requested. These requests are documented with the following PostScript structuring comments:

  ```
  %%DocumentFonts: font1 font2  ...
  ```

- It searches the list of fonts resident on that printer to see if the requested font must be downloaded.

- If the font is not resident on the printer, it searches the host-resident font directory (by getting the appropriate file name from the map table) to see if the requested font is available.
- If the font is available the filter takes the file for that font and prefixes it to the file to be printed.
- The filter sends the font definition file and the source file (the file to be printed) to the PostScript printer.

Customizing the Print Service

Although the LP print service has been designed to be flexible enough to handle most printers and printing needs, it doesn't handle every possible situation. You may buy a printer that doesn't quite fit into the way the LP print service handles printers, or you may have a printing need that the standard features of the LP print service don't accommodate.

You can customize the LP print service in a few ways. This section tells you how you can

- adjust the printer port characteristics,
- adjust the **terminfo** database,
- write an interface program, and
- write a filter.

The diagram in Figure 11–4 gives an overview of the processing of a print request.

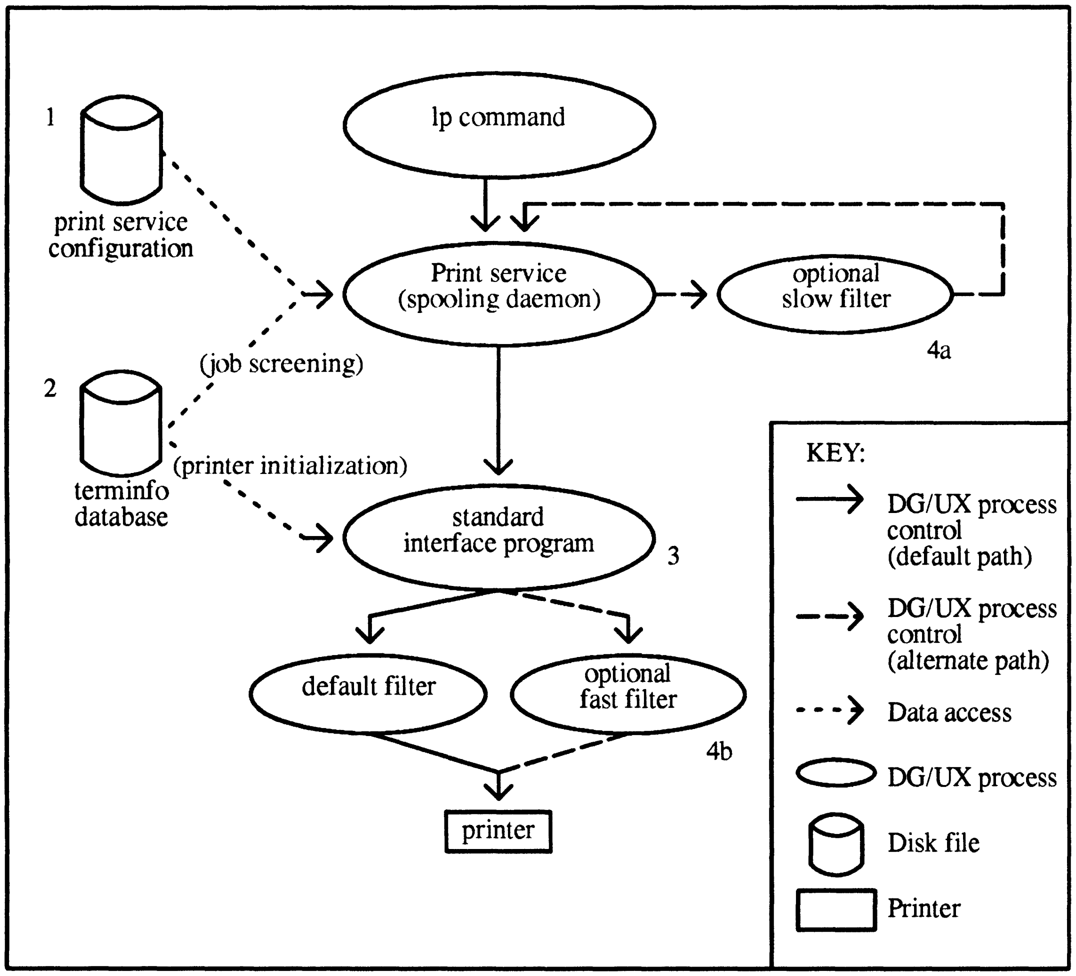

Figure 11–4 How LP Processes Print a Request

Each print request is sent to a spooling daemon that keeps track of all requests. The daemon, which is created when you start the LP print service, is also responsible for keeping track of the status of printers and slow filters; when a printer finishes printing a user's file, the daemon starts it printing another request (if there is one queued).

To customize the print service, adjust or replace some of the pieces shown in Figure 9–6. (The numbers are keyed to the diagram.)

1. For most printers, you need only change the printer configuration stored on disk. The earlier sections of this chapter explain how to do this. Configuration data that are relatively dependent on the printer include the printer port characteristics: baud rate, parity, and so on.

2. For a printer that is not represented in the **terminfo** database, you can add a new entry that describes its capabilities. The **terminfo** database is used in two parallel capacities: screening print requests to ensure that those accepted can be handled by the desired printer, and setting the printer in a state where it is ready to print a request.

   For instance, if the **terminfo** database does not contain an entry for a printer capable of setting a page length requested by a user, the spooling daemon will reject the request. On the other hand, if it does contain an entry for such a printer, then the same information will be used by the interface program to initialize the printer.

3. For particularly difficult printers, or if you want to add features not provided by the delivered LP print service, you can change the standard interface program. This program is responsible for managing the printer: it prints the banner page, initializes the printer, and invokes a filter to send copies of a user's files to the printer.

4a. and 4b.
To provide a link between the applications used on your system and the printers, you can add slow and fast filters. Each type of filter can convert a file into another form, mapping one set of escape sequences into another, for instance, and can provide special setup by interpreting print modes requested by a user. Slow filters are run separately by the daemon, to avoid tying up a printer. Fast filters are run so their output goes directly to the printer; thus they can exert control over the printer.

Adjusting the Printer Port Characteristics

You should make sure that the printer port characteristics set by the LP print service match the printer communication settings. The standard printer port settings have been designed to work with typical files and many printers, but they won't work with all files and printers. This isn't really a customizing step, because a standard feature of the LP print service is to allow you to specify the port settings for each printer. However, it's an important step in getting your printer to work with the LP print service, so it's described in more detail here.

When you add a new printer, read the documentation that comes with it so that you understand what it expects from the host (the LP print service). Then read the manual page for **stty**(1). It summarizes the various characteristics that can be set on a terminal or printer port.

Only some of the characteristics listed in the **stty**(1) manual page are important for printers. The ones likely to be of interest to you appear in Table 11–5 (but you should still consult the **stty**(1) manual page for others).

Table 11–5 stty Options Related to Printers

stty **Option**	**Meaning**
evenp	Send even parity in the 8th bit
oddp	Send odd parity in the 8th bit
–parity	Do not generate parity; send all 8 bits unchanged
110 – 38400	Set the communications speed to this baud rate
ixon	Enable XON/XOFF (also known as START/STOP or DC1/DC3) flow control
–ixon	Turn off XON/XOFF flow control
–opost	Do not do any "output post-processing"
opost	Do "output post-processing" according to the settings listed below
onlcr	Send a carriage return before every linefeed
–onlcr	Do not send a carriage return before every linefeed

Continued

Table 11–5 stty Options Related to Printers

stty Option	Meaning
ocrnl	Change carriage returns into linefeeds
–ocrnl	Do not change carriage returns into linefeeds
–tabs	Change tabs into an equivalent number of spaces
tabs	Do not change tabs into spaces

When you have a set of printer port characteristics you think should apply, adjust the printer configuration as described in "Printer Port Characteristics" under "Configuring Your Printers" in this section. You may find that the default settings are sufficient for your printer.

Adjusting the terminfo Database

The LP print service relies on a standard interface and the **terminfo** database to initialize each printer and establish a selected page size, character pitch, line pitch, and character set. Thus, it is usually sufficient to have the correct entry in the **terminfo** database to add a new printer to the LP print service. Several entries for Data General printers and other popular printers are delivered in the standard **terminfo** database.

Each printer is identified in the **terminfo** database with a short name; this kind of name is identical to the kind of name used to set the **TERM** shell variable. The complete list of supported terminal and printer names are in the terminfo database. To view the entire list, use this command line:

```
# ls –CR  /usr/share/lib/terminfo/*  |  more ⤶
```

If you cannot find a **terminfo** entry for your printer, you should add one. If you do not, you may still be able to use the printer with the LP print service, but you will not have the option of automatic selection of page size, pitch, and character sets, and you may have trouble keeping the printer set in the correct modes for each print request. Another option to follow, instead of updating the **terminfo** entry, is to customize the interface program used with the printer. (See the next section for details on how to do this.)

There are hundreds of items that can be defined for each terminal or printer in the **terminfo** database. However, the LP print service uses fewer than 50 of these. Table 11–6 lists the items that need to be defined (as appropriate for the printer) to add a new printer to the LP print service.

Table 11–6 terminfo Items Relevant to Printers

terminfo Item	Meaning
Booleans:	
cpix	Changing character pitch changes resolution
daisy	Printer needs operator to change character set

Continued

Table 11–6 terminfo Items Relevant to Printers

terminfo Item	Meaning
lpix	Changing line pitch changes resolution
Numbers:	
bufsz	Number of bytes buffered before printing
cols	Number of columns in a line
cps	Average print rate in characters per second
it	Tabs initially every # spaces
lines	Number of lines on a page
orc	Horizontal resolution in units per character
orhi	Horizontal resolution in units per inch
orl	Vertical resolution in units per line
orvi	Vertical resolution in units per inch
Strings:	
chr	Change horizontal resolution
cpi	Change number of characters per inch
cr	Carriage return
csnm	List of character set names
cud1	Down one line
cud	Move carriage down # lines
cuf	Move carriage right # columns
cuf1	Carriage right
cvr	Change vertical resolution
ff	Page eject
hpa	Horizontal position absolute
ht	Tab to next 8-space tab stop
if	Name of initialization file
iprog	Pathname of initializing program
is1	Printer initialization string
is2	Printer initialization string
is3	Printer initialization string
lpi	Change number of lines per inch
mgc	Clear all margins (top, bottom, and sides)
rep	Repeat a character # times

Continued

Table 11–6 terminfo Items Relevant to Printers

terminfo Item	Meaning
rwidm	Disable double wide printing
scs	Select character set
scsd	Start definition of a character set
slines	Set page length to # lines
smgb	Set bottom margin at current line
smgbp	Set bottom margin
smgl	Set left margin at current column
smglp	Set left margin
smgr	Set right margin at current column
smgrp	Set right margin
smgt	Set top margin at current line
smgtp	Set top margin
swidm	Enable double wide printing
vpa	Vertical position absolute

To construct a database entry for a new printer, see details about the structure of the **terminfo** database in the **terminfo**(4) manual page.

Once you've made the new entry, you need to compile it into the database using the **tic**(1) program. Just enter the following command:

```
# tic file-name ↵
```

file–name is the name of the file containing the **terminfo** entry you have crafted for the new printer. The LP print service gains much efficiency by caching information from the **terminfo** database. If you add or delete **terminfo** entries, or change the values that govern pitch settings, page width and length, or character sets, you should stop and restart the LP print service so it can read the new information.

How to Write an Interface Program

If you have an interface program that you used with the LP service before DG/UX Release 5.4, it should still work with the LP print service. Note, though, that several **–o** options have been "standardized," and will be passed to every interface program. These may interfere with similarly named options used by your interface.

If you have a printer that is not supported by simply adding an entry to the **terminfo** database, or if you have printing needs that are not supported by the standard interface program, you can furnish your own interface program. It is a good idea to start with the standard interface program, and change it to fit, rather than starting from scratch. You can find a copy of it under the name **/var/spool/lp/model/standard**.

What Does an Interface Program Do?

Any interface program is responsible for doing the following tasks:

- Initializing the printer port, if necessary. The generic interface program uses the **stty** command to do this.
- Initializing the physical printer. The generic interface program uses the **terminfo** database and the **TERM** shell variable to get the control sequences to do this.
- Printing a banner page, if necessary.
- Printing the correct number of copies of the request content.

An interface program is not responsible for opening the printer port. The LP print service opens the port, a process which includes calling a "dial-up" printer, if one is used to connect the printer. The printer port connection is given to the interface program as standard output, and the printer is identified as the "controlling terminal" for the interface program so that a "hang-up" of the port will cause a **SIGHUP** signal to be sent to the interface program.

A customized interface program must not terminate the connection to the printer or "uninitialize" the printer in any way.

How Is an Interface Program Used?

When the LP print service routes an output request to a printer, the interface program for the printer is invoked as follows:

```
/var/spool/lp/admins/lp/interfaces/P ID user title copies options f1 f2 ...
```

Arguments for the interface program are:

P	printer name
id	request ID returned by the **lp**(1) command
user	username of the user who made the request
title	optional title specified by the user
copies	number of copies requested by the user
options	blank-separated list of options specified by the user or set by the LP print service
f1, f2, ...	full pathnames of the files to be printed

When the interface program is invoked, its standard input comes from **/dev/null**, its standard output is directed to the printer port, and its standard error output is directed to a file that will be given to the user who submitted the print request.

The standard interface recognizes the following values in the blank-separated list in *options*.

nobanner	This option is used to skip the printing of a banner page; without it, a banner page is printed.

nofilebreak This option is used to skip page breaks between separate data files; without it, a page break is made between each file in the content of a print request.

cpi=*decimal–number1*
lpi=*decimal–number2*

These options specify a format of *decimal–number1* columns per inch and *decimal–number2* lines per inch, respectively. The standard interface program extracts from the **terminfo** database the control sequences needed to initialize the printer to handle the character and line pitches.

The words **pica**, **elite**, and **compressed** are acceptable replacements for *decimal–number1* and are synonyms, respectively, for **10** columns per inch, **12** columns per inch, and as many columns per inch as possible.

length=*decimal–number1*
width=*decimal–number2*

These options specify the length and width, respectively, of the pages to be printed. The standard interface program extracts from the **terminfo** database the control sequences needed to initialize the printer to handle the page length and page width.

stty=*'stty–option–list'*

The *stty–option–list* is applied after a default *stty–option–list* as a set of arguments to the **stty** command. The default list is used to establish a default port configuration; the additional list given to the interface program is used to change the configuration as needed.

lpd=*'argument–list'*

This option is used internally by the **lpsched** command; you can ignore it.

flist=*'file–list'* This option is used internally by the **lpsched** command; you can ignore it.

The above options may be specified by the user when issuing a print request. Alternatively, they may be specified by the LP print service from defaults given by the administrator either for the printer (**cpi**, **lpi**, **length**, **width**, **stty**) or for the preprinted form used in the request (**cpi**, **lpi**, **length**, **width**).

Additional printer configuration information is passed to the interface program in the following shell variables:

TERM=*printer–type*

This shell variable specifies the type of printer. The value is used as a key for getting printer capability information from the **terminfo** database.

FILTER=*'pipeline'*

This shell variable specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character–set*

This shell variable specifies the character set to be used when printing the content of a print request. The standard interface program extracts from the **terminfo** database the control sequences needed to select the character set.

A customized interface program should either ignore these options and shell variables or should recognize them and treat them in a consistent manner.

Customizing the Interface Program

Make sure that the custom interface program sets the proper **stty** modes (terminal characteristics such as baud rate and output options). The standard interface program does this, and you can follow suit. Look for the section that begins with the shell comment

```
## Initialize the printer port
```

Follow the code used in the standard interface program. It sets both the default modes and the adjusted modes given by either the LP print service or the user with a line such as the following:

```
stty mode options 0<&1
```

This command line takes the standard input for the **stty** command from the printer port. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below.

```
stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1
```

One printer port characteristic not set by the standard interface program is hardware flow control. The way that this is set will vary, depending on your computer hardware. The code for the standard interface program suggests where this and other printer port characteristics can be set. Look for the section that begins with the shell comment

```
# Here you may want to add other port initialization code.
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. The standard interface program prints a banner that fits on an 80-column page (except for the user's title, which may be longer). Look in the code for the standard interface program for the section that begins with the shell comment

```
## Print the banner page
```

The custom interface program should print all user related error messages on the standard output or on the standard error. The messages sent to the standard error will be mailed to the user; the messages printed on the standard output will end up on the printed page where they can be read by the user when he or she picks up the output.

When printing is complete, your interface program should exit with a code that shows the status of the print job. Exit codes are interpreted by the LP print service as shown in Table 11–7.

Table 11–7 Printer Exit Codes

Code	Meaning to the LP print service
0	The print request has been completed successfully. If a printer fault has occurred, it has been cleared.
1 to 127	A problem has been encountered in printing this particular request (for example, too many non-printable characters, or the request exceeds the printer capabilities). The LP print service notifies the person who submitted the request that there was an error in printing it. This problem will not affect future print requests. If a printer fault had occurred, it has been cleared.

Continued

Table 11–7 Printer Exit Codes

Code	Meaning to the LP print service
128	Reserved for internal use by the LP print service. Interface programs must not exit with this code.
129	A printer fault has been encountered in printing the request. This problem will affect future print requests. If the fault recovery for the printer directs the LP print service to wait for the administrator to fix the problem, the LP print service will disable the printer. If the fault recovery is to continue printing, the LP print service will not disable the printer, but will try printing again in a few minutes.
greater than 129	These codes are reserved for internal use by the LP print service. Interface programs must not exit with codes in this range.

As the table shows, one way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface program exits, the LP print service has no choice but to reprint the request from the beginning when the fault has been cleared. Another way of getting an alert to the administrator (that does not require the entire request to be reprinted) is to have the interface program send a fault message to the LP print service but wait for the fault to clear. When the fault clears, the interface program can resume printing the user's file. When the printing is finished, the interface program can give a zero exit code just as if the fault had never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically, so that the administrator doesn't have to enable the printer.

Fault messages can be sent to the LP print service using the **lp.tell** program. This is referenced using the LPTELL shell variable in the standard interface code. The program takes its standard input and sends it to the LP print service where it is put into the message that alerts the administrator to the printer fault. If its standard input is empty, **lp.tell** does not initiate an alert. Examine the standard interface code immediately after these comments for an example of how the **lp.tell** (LPTELL environment variable) program is used:

```
# Here's where we set up the $LPTELL program to capture
# fault messages.

# Here's where we print the file.
```

If the special exit code 129 or the **lp.tell** program is used, there is no longer a need for the interface program to disable the printer itself. Your interface program can disable the printer directly, but doing so will override the fault alerting mechanism. Alerts are sent only if the LP print service detects the printer has faulted, and the special exit code and the **lp.tell** program are its main detection tools.

If the LP print service has to interrupt the printing of a file at any time, it will terminate the interface program with a signal TERM (signal 15; see **kill**(1) and **signal**(2)). If the interface program dies from receipt of any other signal, the LP print service assumes that future print requests won't be affected, and continues to use the printer. The LP print service notifies the person who submitted the request that the request has not been finished successfully.

When the interface is first invoked, the signals HUP, INT, QUIT, and PIPE (trap numbers 1, 2, 3, and 13) are ignored. The standard interface changes this so that these signals are trapped at appropriate times. The standard interface interprets receipt of these signals as warnings that the printer has a problem; when it receives one, it issues a fault alert.

How to Write a Filter

A filter is used by the LP print service each time it has to print a type of file that isn't acceptable by a printer. A filter can be as simple or as complex as needed; there are only a few external requirements:

- The filter should get the content of a user's file from its standard input and send the converted file to the standard output.
- A **slow** filter can send messages about errors in the file to standard error. A **fast** filter should not, as described below. Error messages from a **slow** filter are collected and sent to the user who submitted the file for printing.
- If a **slow** filter dies because of receiving a signal, the print request is stopped and the user who submitted the request is notified. Likewise, if a **slow** filter exits with a non-zero exit code, the print request is stopped and the user is notified. The exit codes from **fast** filters are treated differently, as described below.
- A filter should not depend on other files that normally would not be accessible to a regular user; if a filter fails when run directly by a user, it will fail when run by the LP print service.

The "Providing Filters" section earlier in this section describes how to add a filter to the LP print service.

If you want your filter to detect printer faults, you must also fulfill the following requirements:

- If possible, the filter should wait for a fault to be cleared before exiting. Additionally, it should continue printing at the top of the page where printing stopped after the fault clears. If the administrator does not want this contingency followed, the LP print service will stop the filter before alerting the administrator.
- It should send printer fault messages to its standard error as soon as the fault is recognized. It does not have to exit, but can wait as described above.
- It should not send messages about errors in the file to standard error. These should be included in the standard output stream, where they can be read by the user.
- It should exit with a zero exit code if the user's file is finished (even if errors in the file have prevented it from being printed correctly).
- It should exit with a non-zero exit code only if a printer fault has prevented it from finishing a file.
- When added to the filter table, it must be added as a **fast** filter. (See "Defining a Filter" in this chapter for details.)

Command Reference for LP Print Service Administration

The commands in Table 11–8 are found in the **/usr/bin** directory unless otherwise noted. To use the administrative commands, you must be the superuser (**sysadm** or **root**).

Table 11–8 LP Print Service Command Reference

Action	Command
Activating a printer:	**enable**(1)
Canceling a request for a file to be printed:	**cancel**(1)
Sending a file (or files) to a printer:	**lp**(1)
Reporting the status of the LP print service:	**lpstat**(1)
Deactivating specified printers:	**disable**(1)
Permitting job requests to be queued for a specific destination:	**accept**(1M)
Preventing jobs from being queued for a specified destination:	**reject**(1M)
Setting up or changing printer configurations:	**lpadmin**(1M)
Setting up or changing filter definitions:	**lpfilter**(1M)
Setting up or changing preprinted forms:	**lpforms**(1M)
Mounting a form:	**lpadmin**(1M)
Moving output requests from one destination to another:	**lpmove**(1M)
Starting the LP print service scheduler:	**lpsched**(1M)
Stop the LP print service scheduler:	**lpshut**(1M)
Setting or changing the default priority and priority limits that can be requested by users of the LP print service:	**lpusers**(1M)

End of Chapter

Chapter 12 Network Management

The DG/UX base system offers three facilities that can provide network service on your system. These facilities, each with its own menu and operations in **sysadm**'s Networking menu, are:

TCP/IP	Loaded as a separate package, the TCP/IP system provides basic Internet network functionality, allowing your system (or host) to communicate with hosts not only on your local network but with virtually any host on the Internet network. TCP/IP provides network file transfer capabilities as well as remote login and remote command execution. TCP/IP also provides security features to regulate access within a network.
NFS/NIS	Loaded as separate packages, the ONC/NFS and NIS systems provide additional functionality based on the TCP/IP software. The ONC/NFS system provides capabilities for sharing file systems on your network. The NIS system, previously known as the Yellow Pages or YP, provides a means of managing DG/UX and TCP/IP databases globally for an entire network, thereby centralizing database management.
UUCP	The UUCP package loads with the DG/UX system. The UUCP system, rather than being based on the TCP/IP software, uses telephone (modem) connections and direct (port-to-port) connections as its network. UUCP offers file transfer and remote command execution capabilities.

Networking Procedures

The following sections discuss the Networking menu in more detail. For a detailed discussion of the Data General TCP/IP package, see *Managing TCP/IP on the DG/UX™ System.* For a detailed discussion of the Data General ONC/NFS and NIS packages, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.* For a detailed discussion of UUCP, see "UUCP Management" later in the chapter.

TCP/IP Management

Managing TCP/IP primarily involves maintaining a lot of databases, files that list various systems, services, addresses, and other network-related information. Consequently, the **sysadm** operations for managing TCP/IP often involve adding database entries, modifying them, deleting them, and listing them. There are other types of operations as well.

If your system is in an NIS domain, you may not need to maintain the TCP/IP databases residing on your system because the NIS system may supply your TCP/IP database needs. If your system is the master server for your NIS domain, you can use the **sysadm** operations to maintain local databases as well as the global databases for the NIS domain. For more information on the NIS package, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

The following sections discuss each TCP/IP database or system feature, describing the purpose and contents of the database, as appropriate. The menus containing the operations used to maintain these databases are in Networking -> TCP/IP.

The coverage of TCP/IP offered here is intended only if you already have some understanding of your TCP/IP network software. For more thorough coverage of the TCP/IP software, see *Managing TCP/IP on the DG/UX™ System.*

Hosts Database

The Hosts database contains the names of hosts in your network, their aliases, and their Internet addresses. The Hosts menu provides operations for adding, deleting, modifying, and listing Hosts entries.

Ethers Database

The Ethers database contains the names of hosts in your network and their Ethernet addresses. An OS server's Ethers database should contain entries for its OS clients. The Ethers menu provides operations for adding, deleting, modifying, and listing Ethers entries.

Networks Database

The Networks database contains the names of networks, the network portion of their Internet addresses, and any alias names assigned to the networks. The Networks menu provides operations for adding, deleting, modifying, and listing Networks entries.

Services Database

The Services database contains the names of services accessible to your network, their aliases, and their addresses. The Services menu provides operations for adding, deleting, modifying, and listing Services entries.

Resolve Address

When the TCP/IP software encounters the name of a host, it must derive the host's Internet address. To match a host name with an address, the TCP/IP system may rely on any or all of three sources: your local Hosts database, which you maintain via the Hosts menu, the global NIS database, and the DNS (Domain Name Server). The TCP/IP software searches these databases in an order that you define with the Search Order operation, below.

The NIS databases are available only if your system is in an NIS domain and has the ONC/NFS/NIS package installed. For more information, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.* The DNS is available only if you have properly configured it on your system. See *Managing TCP/IP on the DG/UX™ System* for more information.

Database Search Order

Use this operation to set the order in which your TCP/IP system searches available resources for the Internet address associated with a host name. The three resources are your local Hosts database, the global NIS database, and the DNS (Domain Name Server).

The Search Order operation lets you specify which database to search first, which to search second, and which to search third. For example, you may want TCP/IP to resolve a host address by scanning the local **hosts** database before resorting to the NIS database and then the DNS.

Resolver Database
If you choose to use the DNS (Domain Name Server) to resolve host names, you need to set the DNS domain name of the resolver. You may also add, delete, modify, and list entries in the resolver database, which contains information on name servers.

Trusted Hosts Database

The Trusted Hosts database (corresponding to the **/etc/hosts.equiv** file) contains the names of users, net groups, and hosts whose users are considered trusted on your system. A trusted user is one who can execute a remote shell, **rcp**, and **rlogin** (without having to supply a password) on your system. The Trusted Hosts menu provides operations for adding, deleting, modifying, and listing entries.

NIS does not support this database. You must maintain it on a host-by-host basis.

SNMP

With SNMP (Simple Network Management Protocol), you can let another host on your network, called a network management station, manage your host's network.

Communities Database
Communities database entries define community names, hosts belonging to the community, and the level of access for the community. The Communities menu provides operations for adding, deleting, modifying, and listing Communities entries.

Traps Database
Traps database entries define hosts, communities, and port numbers that SNMP uses when sending traps (special asynchronous messages) to network management stations. The Traps menu provides operations for adding, deleting, modifying, and listing Traps entries.

Objects Database
Values in the Objects database override default values supplied by the SNMP daemon. The Objects menu provides operations for setting, getting, resetting, and listing values.

Host Name/Host ID

The Hostname/Hostid menu provides operations for setting your host's host name and host ID. The host name is the name returned by the **hostname**(1C) command. The host ID is a hexadecimal representation of the Internet address associated with the host name.

Interfaces

Your network interfaces are the boards or devices your host uses to connect to a network. Each interface is associated with an entry in the **/dev** directory. The Interfaces menu provides operations for adding, deleting, modifying, and listing network interfaces as well as for starting and stopping them.

For a newly-added network interface to work, you may have to rebuild your kernel to include the appropriate driver (such as **hken()** or **ixe()**, for example). You do not need to rebuild your kernel if the system file used to build your kernel contains the correct device driver entry. If the proper device driver is configured in your kernel, the interface will start when next you boot the kernel. For more information on building kernels, see Chapter 4.

Routes

To contact hosts not on your immediate local network, your system needs routing information telling how to find the other host. The Routes menu provides operations for adding, deleting, modifying, and listing route entries. You can specify whether you want the route to be permanent (available every time you start the network) or temporary (available until the next time the system or network software stops).

For more information on routing, see *Managing TCP/IP on the DG/UX™ System.*

Daemons

There are a number of daemons, programs that run constantly in the background, that manage network services. This menu provides operations for adding, deleting, modifying, listing, starting, and stopping daemons.

Network daemons fall into two categories: those run by the **inetd** daemon and those independent of **inetd**. Daemons run by **inetd** include **ftpd**, **telnetd**, **rshd**, and others appearing in **/etc/inetd.conf**. The **inetd** daemon manages the starting and stopping of its daemons as appropriate. Daemons independent of **inetd** include **smtp**, **rwhod**, **pmtd**, **snmpd**, and others listed in **/etc/tcpip.params**. Typically, your system's **rc** scripts (discussed in Chapter 3) manage starting and stopping these daemons. In addition, modifying an independent daemon's parameters stops and restarts it.

Shell Names

Historically, there has been a conflict in the naming conventions that apply to the restricted shell and the remote shell. The Shell Names menu provides operations that you may use to set and get these shell names. You can choose to make your system conform to the AT&T System V Release 4 conventions, or you can choose to conform to the BSD conventions.

If you choose to make your system conform to the System V Release 4 conventions, the operation names the shells like this:

Restricted shell:	**/bin/rsh** and **/bin/restsh**
Remote shell:	**/usr/bin/remsh**

If you choose to make your system conform to the BSD conventions, the operation names the shells like this:

Restricted shell:	**/bin/restsh**
Remote shell:	**/usr/bin/rsh** and **/usr/bin/remsh**

NFS/NIS Management

The ONC/NFS system allows hosts on your local network to share file systems as though they resided locally. This NIS system provides tools for managing DG/UX and TCP/IP databases through a single host acting as a master NIS database server.

The **sysadm** Networking –> NFS/NIS –> Parameters menu allows you to set and get the following parameters that govern how ONC/NFS and NIS operate. The parameters appear on your system in **/etc/nfs.params**.

NIS Domain Name

An NIS domain is a group of systems using NIS to manage a common set of DG/UX and TCP/IP databases. Your NIS administrator chooses the domain name. In **/etc/nfs.params**, this parameter appears as **domainname_ARG**.

NIS Role

A member of an NIS domain can serve in one of these capacities:

MASTER	The system in the domain where the NIS administrator maintains the global NIS databases. The master system copies the NIS databases on a regular basis to selected systems in the domain that act as servers, below.
SERVER	One of several selected systems in the domain that receives copies of the NIS databases from the master. The server provides access to these databases to clients, below.
CLIENT	Any system in the domain that does not have its own copies of the NIS databases. When a client starts its network software, it associates itself with an NIS server in its domain. Thereafter, when the client needs any information from the NIS databases, it retrieves it from the server.

In **/etc/nfs.params**, this parameter appears as **ypserv_START**.

Start NFS Daemon Services

There are various daemons associated with ONC/NFS services, such as **portmap**, **nfsd**, **mountd**, and others appearing in **/etc/nfs.params** or **/usr/sbin/init.d/rc.nfsserv**. You can set a parameter to start these daemons when the network software starts (by default, at run level 3). Otherwise, you must start the daemons yourself. If you intend to mount remote file systems using ONC/NFS, set this parameter to start daemons.

In **/etc/nfs.params** this parameter appears as **nfsserv_START**, and the ONC/NFS daemons appear in the parameter **NFSSERV_DEMONS.**

Mount Remote NFS File Systems

This parameter determines whether or not your system will mount remote file systems when you start network software (typically at boot time). The ONC/NFS system mounts any file systems in the file system table (**/etc/fstab**) that are type **nfs**. In **/etc/nfs.params** this parameter appears as **nfsfs_START**.

Number of nfsd Daemons

The **nfsd** daemon handles requests from remote hosts that want to access your local file systems. The number of daemons you start depends on the amount of remote access you expect. In **/etc/nfs.params** this parameter appears as **nfsd_ARG**.

Arguments to statd

These are any additional arguments that you want to pass to **statd** when starting it at system boot. See the **statd**(1M) manual page for more information. In **/etc/nfs.params** this parameter appears as **statd_ARG**.

Arguments to lockd

These are any additional arguments that you want to pass to **lockd** when starting it at system boot. See the **lockd**(1M) manual page for more information. In **/etc/nfs.params** this parameter appears as **lockd_ARG**.

Time to delay after lockd

This is the number of seconds delay that will occur after the system starts the **lockd** daemon when the system comes up to run level 3. This delay gives the **lockd** daemon a chance to establish connections with other **lockd** daemons on the network. The default is 15. If your system does not export any file systems for remote access, you may specify 0. In **/etc/nfs.params** this parameter appears as **lockd_sleep_ARG**.

Arguments to mountd

The system passes these arguments to **mountd**(1M) when it starts **mountd** at boot. See the **mountd**(1M) manual page for more information. In **/etc/nfs.params** this parameter appears as **mountd_ARG**.

See *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for more information.

UUCP Management

The UUCP system uses telephone lines and direct port-to-port connections as its network rather than Ethernet and Internet networks. Like TCP/IP, it functions using several daemons and a number of database files. For complete coverage of UUCP, including more detailed discussions of the configuration files, see "UUCP Management" in this chapter.

The following sections discuss the **sysadm** Networking –> UUCP menus and operations in more detail.

Devices

This database, **/etc/uucp/Devices**, contains entries for the dial-out devices, or modems, used to connect with remote systems. Each Devices entry contains fields for the device location (port pathname in the **/dev** directory) and line speed of the attached ACU (automatic calling unit, or modem), if used, or information about other types of connecting devices.

The Devices menu provides operations for adding, deleting, modifying, and listing Devices entries.

Polling

This database, **/etc/uucp/Poll**, contains entries for the remote systems that your system needs to call to initiate UUCP transfers. Polling is necessary for contacting sites that do not have hardware, such as modems, necessary for calling out to other systems. Each Poll entry contains fields for the remote system name and the times when calling should take place.

The Polling menu provides operations for adding, deleting, modifying, and listing Poll entries.

Systems

This database, **/etc/uucp/Systems**, contains entries for remote hosts that you want to contact using UUCP. Each entry contains fields for the remote system name, the name of the device used to connect to the remote system, the times during which you may reach the remote system, the telephone number of the remote system's modem (if access is via modem), and login information.

The Systems menu provides operations for adding, deleting, modifying, and listing Systems entries.

Test

This operation tests connections to remote systems. After you supply a remote system name, UUCP uses Systems and Devices information to attempt to establish a connection with the remote system.

UUCP Management

The DG/UX System uses the HoneyDanBer version of UUCP. You can find additional expert information in "Expert UUCP Information" at the end of the chapter. For a list of UUCP error messages and explanations, see Appendix A.

What Is UUCP?

UUCP, which stands for UNIX-to-UNIX copy, refers to a set of programs and data files that allow you to transfer files and to execute remote commands between UNIX systems. In this section, the name UUCP refers to the entire system of programs and databases; the name **uucp** refers to the **uucp**(1) command itself.

When we talk about UUCP connections, we mean via a *direct* or a *dial-up* connection. A direct connection is simply a physical wire between machines. A dial-up connection uses telephone lines and a modem at both ends to connect machines.

Understanding the UUCP Components

The UUCP system consists of multiple files and programs. The five main configuration files, located in **/etc/uucp**, are **Systems**, **Poll**, **Devices**, **Dialers**, and **Permissions**. You can connect only to the remote systems listed in **Systems**. You can do this from the command line for an immediate connection, or you can connect and transfer files automatically at the times set in the **Poll** file. The **uudemon.poll** shell script reads the **Poll** file and initiates connections. Files queued for transfer are exchanged via the modems and devices listed in the **Devices** file. The entries in **Devices** need data from **Dialers**. Finally, the **Permissions** file restricts a remote host's ability to request and receive files. The default **Permissions** file is set up to provide the maximum amount of security. You can use the **uucheck –v** command to see exactly what your default permissions are. If you wish to change them, see "Permissions File" in "Expert UUCP Information" at the end of the chapter.

Before you can put this system to work, you must have a location with which you wish to set up file transfer connections. This means you will have to contact the system administrator of a remote site and exchange certain information: passwords, system NODE names, baud rates, and phone numbers. With this information, you are ready to set up the UUCP files.

UUCP Setup Overview

We recommend that you set your UUCP facility up in the following order:

1. Read "HoneyDanBer: New UUCP versus Old UUCP."
2. Start the **uudemon.poll**, **uudemon.hour**, **uudemon.admin**, and **uudemon.cleanup** shell scripts.
3. Add devices with the operation Networking –> UUCP –> Devices –> Add.
4. Add systems with the operation Networking –> UUCP –> Systems –> Add.
5. Set up lines for dialing in and out with the operation Devices –> Port –> Port Services –> Add.
6. Add poll entries with the operation Networking –> UUCP –> Polling –> Add .
7. Test your connections with the operation Networking –> UUCP –> Test.

HoneyDanBer: New UUCP versus Old UUCP

There are two versions of UUCP: a newer version, referred to as HoneyDanBer, and an older version. Functional differences between the two versions of UUCP are reflected in the **/etc/inittab** file. The HoneyDanBer version allows for bidirectional login by adding a bidirectional port service (formerly implemented by respawning **uugetty** instead of **getty**). For more information on port services and the SAF facility, see Chapter 9.

A bidirectional port service can call as well as receive. DG/UX UUCP supports connections with systems running old UUCP. To enable such a connection, you need to set up the **inittab** and **Systems** files correctly. After using the **sysadm** operation Networking –> UUCP –> Systems –> Add, you will need to edit some of the data files to reflect what versions of UUCP you will communicate with. See "Connecting Like and Unlike Versions of UUCP" later in this chapter for more information.

Starting the UUCP Shell Scripts

Your first step in setting up UUCP is to start several important shell scripts:

uudemon.poll Reads the **Poll** file (**/etc/uucp/Poll**) as scheduled.

If any of the systems in the **Poll** file are scheduled to be polled, places a work file (C.*file_name*) in the directory **/var/spool/uucp/***node-name* (where *node-name* is the name of the system to be polled).

uudemon.hour Calls the **uusched** program to search the spool directories for work files (**C.***file_name*) that have not been processed, and schedules these files for transfer to a remote machine.

Calls the **uuxqt** daemon to search the spool directories for execute files (**X.***file_name*) that have been transferred to your host and were not processed at the time they were transferred.

uudemon.admin Runs the **uustat** command with **–p** and **–q** options. The **–q** option reports on the status of work files (**C.***file_name*), data files (**D.***file_name*), and execute files (**X.***file_name*) that are queued. The **–p** prints status information for processes listed in **/var/spool/locks**.

Sends resulting status information to the **nuucp** administrative login via **mail**(1).

uudemon.cleanup Takes log files for individual machines from the directory **/var/spool/uucp/.Log**, merges them, and places them in the directory **/var/spool/uucp/.Old** with other old log information.

Removes from the spool directory work files seven days old or older, data files seven days old or older, and execute files two days old or older.

Returns mail that cannot be delivered to the sender.

Mails a summary of the status information gathered during the current day to the **nuucp** administrative login.

To run the scripts on a regular basis, schedule them as **cron** jobs to be run under the username **nuucp**. Use the **su**(1) command to change your username to **nuucp**. Then use **crontab** to schedule the UUCP jobs. The recommended **cron** jobs for UUCP are in the file **/admin/crontabs/uucp.proto**. See "Automating Job Execution" in Chapter 2 for more information on **cron**.

UUCP Programs, Daemons, and Data Files

This section discusses the major components of UUCP

If you administer UUCP without **sysadm**, use the **nuucp** login ID because it owns the UUCP files and the spooled data files. The other UUCP login ID is **uucp**, which UUCP systems on remote hosts use when they need to transact UUCP business with another machine. Instead of starting a shell like a normal user's login, the **uucp** profile starts the **uucico** program. See Chapter 4 for more information on administrative logins.

Administrative Programs

The following administrative programs are available for use for your convenience.

uuname Lists those machines you can contact. This command is in **/usr/bin**. You can do this from the command line or you can use the Systems menu's List operation.

uulog	Displays the contents of the log directories for specified hosts. Log files are created for each remote host with which your host communicates. The log files contain records for each use of **uucp**, **uuto**, and **uux**. This command is in **/usr/bin** and is not available through **sysadm**.
uucleanup	Cleans up the spool directory. This command is normally executed from a shell script called **uudemon.cleanup**, which is started by the **cron** facility.
uutry	Tests connections between hosts. This command displays messages on failed or successful sessions and does a moderate amount of debugging. It invokes the **uucico** daemon to establish a communication link between your host and the remote host you specify. You can also do this with the UUCP menu's Test operation.
uucheck	Checks for the presence of UUCP directories, programs, and support files. With a **–v** option, it displays the current permissions for your system. This command is in **/usr/lib/uucp** and is not available through **sysadm**.

User Programs

The user programs for UUCP are in **/usr/bin**. No special permission is needed to use these programs.

ct	This program instructs your host to call a modem attached to a terminal over the telephone network. When the modem answers, the SAF facility starts the **login** port service for the modem and allows the terminal user to log in. The user of the remote terminal may call the host and request that it call the remote terminal back. The host will hang up the initial link to the terminal so that it will be available to answer the call back. This is similar to making a collect call.
cu	This program connects your host to a remote host and allows you to be logged in on both hosts at the same time. You can execute commands on either host without dropping the communication link.
uucp	This program lets you copy files from one host to another. It creates work files and data files, queues the job for transfer, and calls the **uucico** daemon, which in turn attempts to contact the remote host.
uuto	This program copies files from one host to a public spool directory on another host (**/var/spool/uucppublic/receive**). Unlike **uucp**, which lets you copy a file to any accessible directory on the remote host, **uuto** places the file in an appropriate spool directory and tells the remote user to pick it up with the **uupick** program.
uupick	This program retrieves the files placed under **/var/spool/uucppublic/receive** when files are transferred to a host using **uuto**.
uux	This program creates the work, data, and execute files needed to execute commands on a remote host. The work file contains the same information as work files created by **uucp** and **uuto**. The execute files contain the command string to be executed on the remote host and a list of the data files. The data files are those files required for the command execution.

Licensed Material – Property of copyright holder(s)

uustat	This program displays the status of requested file transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Daemons

Daemons are routines that run as background processes and perform system-wide public functions. These daemons handle file transfers and command executions. They can also be run manually from the shell.

uucico	Selects the device used for the link, establishes the link to the remote host, performs the required login sequence and permission checks, transfers data and execute files, logs results, and notifies the user by mail of transfer completions. When the local **uucico** daemon calls a remote system, it interacts with the **uucico** daemon on the remote system during the session. The **uucico** daemon is executed by **uucp**, **uuto**, and **uux** programs, after all the required files have been created, to contact the remote host. It is also executed by the **uusched** and **uutry** programs.
uuxqt	Executes remote execution requests. The **uuxqt** daemon searches the spool directory for execute files (**X.***file*) that have been sent from a remote host. When an execute file is found, **uuxqt** opens it to get the list of data files that are required for the execution. It then checks if the required data files are available and accessible. If the files are present and can be accessed, **uuxqt** checks the **Permissions** file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed by the **uudemon.hour** shell script, which is started by the **cron** facility.
uusched	Schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote hosts will be called. The **uusched** daemon is executed by a shell script called **uudemon.hour**, which is started by the **cron** facility.

UUCP Data Files

The support files for UUCP are in the directory **/etc/uucp**. You can make all changes to these files through **sysadm**'s UUCP menu. Below, we provide details on the structure of these files in case you want to edit them manually.

Devices	Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices.
Dialers	Contains character strings required to negotiate with network devices (automatic calling devices) to establish connections to remote hosts (non-801-type dialers). This file contains some sample chat scripts. A chat script is a series of modem commands and strings that **uucico** sends to a modem to initiate a connection (or attempt to connect) with a remote system.
Systems	Contains information needed by the **uucico** daemon and the **cu** program to establish a link to a remote host. It contains information such as the name of the remote host, the name of the connecting device associated with the remote host, when the host can be reached, telephone number, login ID, and password.

Dialcodes	Contains dial-code abbreviations that may be used in the phone number field of **Systems** file entries.
Permissions	Defines the level of access that is granted to hosts when they attempt to transfer files or remotely execute commands on your host.
Poll	Defines hosts that are to be polled by your system and when they are polled.
Sysfiles	Assigns different or multiple files to be used by **uucico** and **cu** as **Systems**, **Devices**, and **Dialers** files.

UUCP Directories

This section lists and describes the directories necessary to run UUCP.

/usr/bin	Contains UUCP user programs.
/usr/lib/uucp	Contains the executable files for the UUCP system.
/usr/spool/uucp	The HOME directory for the **uucp** login.
/etc/uucp	Contains the files that make up the UUCP database.
/var/spool/locks	Contains lock files for UUCP devices.
/var/spool/uucp	Contains directories for administrative purposes and for storing log and status information. The spool directory for queued work that is to be processed by UUCP daemons.
/var/spool/uucppublic	Stores work that has been sent to your host. The public directory for UUCP.

Remedies for Common UUCP Problems

This section contains remedies for some of the common problems that may prevent UUCP from operating correctly.

Remote system down

- Call the remote system's number yourself and listen for the high-pitched tone of the answering modem. If there is none, you know the system or modem is not operating. Call the system administrator of the remote system.

Incorrect login information for remote system

- Dialing sequence: look in the **Systems** and **Dialcodes** files. Consider inserting pauses (commas) in the dialing sequence.
- Login name/password: make sure the login name/password in the local **Systems** file match the login name/password in the remote system's **passwd** file.
- Login sequence: examine the expect/send sequence in the **Systems** file and make sure it uses the correct conventions and login/password strings.
- Remote system name mismatch: make sure the system name in the **Systems** entry matches the nodename of the remote system. On the remote system, use the **uname –n** command to get the nodename.

Licensed Material – Property of copyright holder(s)

Modem cannot make connection

- Verify that your modem switches are properly set (refer to modem documentation). Read the **Dialers** file for guidance.
- Make sure your **Dialers** file is set correctly for touch tone or pulse.

Cannot dial in

- Verify that your modem switches are correct.
- Put a phone on the line and call the remote system's modem.
- Make sure that the desired **ttymon** service is running on the port to which the modem is attached. Set up port services with the operation Device –> Port –> Port Service –> Add.
- If you set the port for bidirectional use, try typing anything or press New Line.

uucico or cu dies immediately

- Use appropriate debugging switch first (for **uucico**(1M), **–x**; for **cu**(1), **–d**). If you get no response or a hangup, make sure **Systems**, **Devices**, **Permissions**, and **Dialers** files are present and readable by **uucp**.
- Make sure **passwd** and **group** files are correct on both systems.

Hung modem

- Send the reset command (such as ATZ for Hayes modems) to the modem. You can do this with the **cu**(1) command or by turning the modem on and off.

Hung syac

- Reload the **syac** with **/usr/sbin/tcload**.

Modem in wrong state (such as echo mode)

- Examine the modem switches. In the case of Hayes-compatible modems, sending ATZ to the modem may fix the problem. See the documentation for your modem. You can find some modem settings in the **Dialers** file.

Getty on line

- Traditionally, you had to put a **getty** on a dial-in line but not on a dial-out line. For a bidirectional port, **uugetty** had to be used. DG/UX Release 5.4 and later systems use the SAF facility instead of **getty** and **uugetty**. On a DG/UX Release 5.4 or later system, you set up **login** port service, non-bidirectional, for a dial-in-only line; you set up no port service for a dial-out-only line; and you set up **login** port service, bidirectional, for a line that you can use both ways. See the Port menu of **sysadm**'s Device menu. Chapter 9 covers the Port menu.

Wrong line speed

- Make sure the modem line speed (baud rate) is compatible with the entry in the **/etc/inittab** file.

Wrong permissions/access

- The device file (such as **/dev/tty***xx*) should permit reading and writing for any user. Permissions should be set at 666. All files in **/etc/uucp**, **/usr/lib/uucp**, and **/var/spool/uucp** should be owned by **uucp**.

Too many unsuccessful attempts

- Remove the status files (files with names matching **/var/spool/uucp/.Status/***system_name*, where *system_name* is the name of the remote system in the UUCP request). Also remove any related lock files (**/var/spool/locks/LCK.***) and start **uucico** yourself to try and complete the job. You may also have to delete any temporary files (**/var/spool/uucp/***system_name***/TM***, where *system_name* is the name of the remote system in the UUCP request) that **uucico** might have created during a file transfer.

No daemon

- Make sure the **uucp** or **uux** commands start the **uucico** or **uuxqt** daemons without problems.

Permissions file

- Make sure this file does not prohibit the desired transfer. Make sure it contains an entry for the **uucp** login name on the remote system.

Bad modem/phone line

- Test the modem and cable and make sure they are functioning properly. Test the phone line for noise or interruptions.

Out of file system space

- There is not enough space on the remote system to transfer the file. Contact the administrator of the remote system.

Bad pathname

- Incorrect or illegal pathname. Enter the correct pathname.

Expert UUCP Information

This section gives further information on some of the topics covered earlier in the chapter.

UUCP Connections

Before your system can communicate with a remote system, you must set up a two-way communication connection between the systems. This section describes the two kinds of UUCP connections: the direct connection and the dial-up connection.

Direct Connection

The direct connection communication method requires a direct connection from a port on a local host to a port on the remote host. A direct line is advantageous when communication is required with the remote host on a regular basis. The link is always available and access time is short. The disadvantage of the direct link is that the port cannot be used for anything else. The direct connection is made over an RS-232C serial port at transmission rates of up to 19200 bits/second. The recommended length of direct links is 50 feet (around 15.5 meters) or less. Longer lengths can be obtained by using a lower transmission rate and/or limited distance modems at both ends of the link.

Direct connections are beneficial only when:

- It is not possible to link the hosts together through a local area network (LAN).
- Two hosts transfer large amounts of data on a regular basis.
- Two hosts are connected by no more than several hundred feet of cable.

The distance between two directly-linked hosts is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet (around 15.5 meters) or less with transmission rates as high as 19200 bits per second. As the cable length is increased, noise on the lines may become a problem, which means that the transmission rate must be decreased or limited distance modems be placed on each end of the line.

Do not use more than 1000 feet of cable to connect the two hosts or communications will be unreliable. This link should operate comfortably at 9600 bits per second in a clean (noise free) environment.

Dial-up Connection

In this case, the host that is going to make the connection would call the remote host using an Automatic Calling Unit (ACU). The remote host answers via its own ACU and makes the connection. With this arrangement, the ports are not dedicated to only one host. A dial-up link also requires more hardware (such as the ACU) than the direct connection. Transmission rates are limited to the capacity of the ACUs.

Refer to your modem documentation for information on configuring your modem for dial-out or dial-in use. In **/etc/uucp/Dialers.proto**, you'll find a description for setting up a Hayes modem.

If your modem name is not listed in the **Dialers** file, you will need to edit this file and create a chat script with your modem name as a label. The chat script is the sequence of commands a modem uses for dialing out. Refer to your modem documentation for information on your modem's language and command syntax.

Modem and Direct Link Support Files

Be sure to update the following support files to reflect the presence of a direct link or a modem connection:

- **/etc/uucp/Devices**
- **/etc/inittab**
- **/etc/uucp/Systems**

Additionally, the **/etc/uucp/Dialers** file must contain information on any modem you use.

When you have determined which communication links best suit your needs, you will need to set up port services for those lines. To set up tty lines and port services, see the Port menu in **sysadm**'s Device menu. Chapter 9 discusses the Port menu.

UUCP Data Files

UUCP data files must be owned by **nuucp** and must have Read and Write access permissions. The following sections describe the files in **/etc/uucp** that support UUCP file transfers. The names of files discussed are:

Devices	**Dialers**
Systems	**Dialcodes**
Permissions	**Poll**
Sysfiles	**Maxuuxqts**
Maxuuscheds	**remote.unknown**

Devices File

The **Devices** file (**/etc/uucp/Devices**) contains information for all the devices that may be used to establish a link to a remote host; these are devices such as automatic call units, direct links, and network connections.

NOTE: This file works closely with the **Dialers**, Systems, and **Dialcodes** files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the **Devices** file has the following format:

```
Type Line Line2 Class Dialer-Token-Pairs...
```

Each of these fields is defined in the following section.

Type This field may contain one of two keywords (**Direct** or **ACU**), the name of a local area network switch, or a system name.

Direct	This keyword indicates a Direct Link to another host or a switch (for **cu**(1) connections only).
ACU	This keyword indicates that the link to a remote host is made through an automatic call unit (also known as an automatic dial modem). This modem may be connected either directly to your host or indirectly through a LAN switch.
LAN_Switch	This value can be replaced by the name of a LAN switch. Micom and Develcon are the only LAN switches for which there are caller scripts in the **Dialers** file. You can add your own LAN switch entries to the **Dialers** file.
Sys–Name	This value indicates a direct link to a particular host. (*Sys–Name* is replaced by the name of the host.) This naming scheme is used to

convey the fact that the line associated with this **Devices** entry is for a particular host in the **Systems** file.

The keyword used in the *Type* field is matched against the third field of **Systems** file entries as shown below:

Devices: `ACU tty11 - 1200 penril`

Systems: `eagle Any` **ACU** `1200 3251 ogin: uucp sword: Oakleaf2`

Line This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the ACU for a particular entry was attached to the **/dev/tty11** line, the name entered in this field would be **tty11**.

Line2 If the keyword **ACU** appears in the *Type* field and the ACU is an 801 type dialer, *Line2* would contain the device name of the 801 dialer. (801 type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line, defined in the *Line* field.) This means that one line would be allocated to the modem and another to the dialer. Since non-801 dialers will not normally use this configuration, the *Line2* field will be ignored by them, but it must still contain a hyphen (–) as a placeholder.

Class If the keyword **ACU** or **Direct** is used in the *Type* field, *Class* may be just the speed of the device. However, it may contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). The letter is necessary because many larger offices may have more than one type of telephone network: one network may be dedicated to serving only internal office communications while another handles the external communications. In such a case, it becomes necessary to distinguish which lines should be used for internal communications and which should be used for external communications. The keyword used in the *Class* field of the **Devices** file is matched against the fourth field of **Systems** file entries as shown below:

Devices: `ACU tty11 -` **D1200** `penril`

Systems: `eagle Any ACU` **D1200** `3251 ogin: nuucp sword: Oakleaf2`

Some devices can be used at any speed, so the keyword **Any** may be used in the *Class* field. If **Any** is used, the line will match any speed requested in a **Systems** file entry. If this field is **Any** and the **Systems** file *Class* field is **Any**, the speed defaults to 1200 bps.

Dialer-Token-Pairs

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, or it may be **direct** for a Direct Link device. You can have any number of *Dialer-Token-Pairs*. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the **Systems** file.

This field has the format:

dialer token dialer token

The last pair of tokens may or may not be present, depending on the associated device (dialer). In most cases, the last pair contains only a *dialer* portion. The *token* portion is retrieved from the *Phone* field of the **Systems** file entry.

A valid entry in the *dialer* portion may be defined in the **Dialers** file or may be a special dialer type. The 801 – Bell 801 auto dialer is compiled into the software and is therefore available without having an entry in the **Dialers** file.

```
801 - Bell 801 auto dialer
```

The *Dialer-Token-Pairs* (*DTP*) field may be structured four different ways, depending on the device associated with the entry. Note that any **\T** or **\D** escape sequence describes the token *but is not the token*. See below.

1. If an automatic dialing modem is connected directly to a port on your host, the *DTP* field of the associated **Devices** file entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** file entry with an entry in the **Dialers** file. Therefore, the *dialer* field must match the first field of a **Dialers** file entry as shown below:

   Devices: `ACU tty11 - 1200` **ventel**

   Dialers: **ventel** `=&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!`

   Notice that only the *dialer* portion (**ventel**) is present in the *DTP* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry.

2. If a direct link is established to a particular host, the *DTP* field of the associated entry would contain the keyword **direct**. This is true for both types of direct link entries, **Direct** and *System-Name* (refer to discussion on the *Type* field).

3. If a host with which you wish to communicate is on the same local network switch as your host, your host must first access the switch and the switch can make the connection to the other host. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry as shown below:

   Devices: `develcon tty13 - 1200` **develcon** `\D`

   Dialers: **develcon** `"" "" \pr\ps\c est:\007 \E\D\e \007`

   As shown, the *token* portion is left blank, which indicates that it is retrieved from the **Systems** file. The **Systems** file entry for this particular host will contain the token in the *Phone* field, which is normally reserved for the phone number of the host (refer to **Systems** file, *Phone* field). This type of *DTP* contains an escape character (**\D**), which ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the **Dialcodes** file.

4. If an automatic dialing modem is connected to a switch, your host must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the **Dialers** file as shown below:

   Devices: `ACU tty14 - 1200` **develcon** `vent` **ventel**

   Dialers: **develcon** `"" "" \pr\ps\c est:\007 \E\D\e \007`
   Dialers: **ventel** `=&-% "" \r\p\r\c $ <K\eT%%\r>\c ONLINE!`

Licensed Material – Property of copyright holder(s)

In the first pair, **develcon** is the dialer and **vent** is the token that is passed to the Develcon switch to tell it which device (**ventel** modem) to connect to your host. This token would be unique for each LAN switch since each switch may be set up differently. Once the **ventel** modem has been connected, the second pair is accessed, where **ventel** is the dialer and the token is retrieved from the **Systems** file.

There are two escape characters that may appear in a *DTP* field:

\T Indicates that the *Phone* (*token*) field should be translated using the **Dialcodes** file. This escape character is normally placed in the **Dialers** file for each caller script associated with an automatic dial modem (**penril**, **ventel**, and so on). Therefore, the translation will not take place until the caller script is accessed.

\D Indicates that the *Phone* (*token*) field should not be translated using the **Dialcodes** file. If no escape character is specified at the end of a **Devices** entry, the **\D** is assumed (default). A **\D** is also used in the **Dialers** file with entries associated with network switches (develcon and micom).

Dialers File

The **Dialers** file (**/etc/uucp/Dialers**) specifies the initial conversation that must take place on a line before it can be made available for transferring data. This conversation is usually a sequence of ASCII strings that are transmitted or expected (called a chat script). A chat script is often used to dial a phone number using an ASCII dialer (such as an automatic dial modem).

As shown earlier, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801). Here an attempt is made to match the fifth field in the **Devices** file with the first field of each **Dialers** file entry. In addition, each odd numbered **Devices** field (the token field) starting with the seventh position is used as an index into the **Dialers** file. If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations.

Each entry in the **Dialers** file has the following format:

```
dialer substitutions expect-send ...
```

The *dialer* field matches the fifth and additional odd numbered fields in the **Devices** file. The *substitutions* field is a translation string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate the equal (=) and hyphen (–) characters into whatever codes the dialer requires for "wait for dialtone" and "pause."

The remaining *expect-send* fields are character strings. Below are some character strings distributed with the **Dialers** file.

```
penril =W-P "" \d > K\c : \EP\T OK
penril_old =W-P "" \d > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c
OK
ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
hayes =,-,  "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
hayes_att =,-, "" \dAT\r\c OK\r ATDT\T\r\c CONNECT
rixon =&-%  "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadic =K-K "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
develcon "" "" \pr\ps\c est:\007 \E\D\e \007
micom ""     "" \s\c NAME? \D\r\c GO
direct
```

```
att2212c      =+-,    "" \r\c :--: atol2=y,T\T\r\c red
att4000 =,-,  "" \033\r\r\c DEM: \033s0401\c \006 \033s0901\c \
        \006 \033s1001\c \006 \033s1102\c \006 \033dT\T\r\c \006
att2224 =+-,    "" \r\c :--: T\T\r\c red
nls     ""      "" NLPS:000:001:1\N\c
```

Three AT&T modems have entries in the **Dialers** file. The Penril, Micom modem, and Hayes modem scripts have all been confirmed at Data General as have the Micom and Develcon data switches. The other entries have not been tested. If you need to modify the supplied script, refer to your modem documentation.

Table 12–1 shows the meanings of some of the escape characters (those beginning with ****) used in the **Dialers** file.

Table 12–1 Escape Characters Used in the Dialers File

Escape Character	Description
\D	Phone number or token without **Dialcodes** translation.
\E	Enable echo checking (for slow devices).
\T	Phone number or token with **Dialcodes** translation.
\K	Insert a BREAK.
\c	No New Line or carriage return.
\d	Delay (approximately 2 seconds).
\e	Disable echo checking.
\n	Send New Line.
\p	Pause (approximately 1/4 to 1/2 second).
\r	Carriage return.
nnn	Send octal number *nnn*.

Additional escape characters that may be used are listed in the section discussing the **Systems** file.

The Penril entry in the **Dialers** file is executed as follows. First, the phone number argument is translated, replacing any equal sign (=) with a `W` (wait for dialtone) and replacing any hyphen (–) with a `P` (pause). The handshake given by the remainder of the line works as follows:

`""`	Wait for nothing. (In other words, proceed to the next thing.)
`\d`	Delay for 2 seconds.
`>`	Wait for a >.
`K\c`	Send a `K`. Send no terminating New Line.
`:`	Wait for a `:`.

`\EP\T` Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send a `P` and the phone number. The `\T` means take the phone number passed as an argument and apply the **Dialcodes** translation and the modem function translation specified by field 2 of this entry.

`OK` Waiting for the string `OK`.

Systems File

The **Systems** file (**/etc/uucp/Systems**) contains the information needed by the **uucico** daemon to establish a communication link to a remote host. Each entry in the file represents a host that can be called by your host. In addition, UUCP software can be configured to prevent any host that does not appear in this file from logging in on your host. More than one entry may be present for a particular host. The additional entries represent alternative communication paths that will be tried in sequence.

Using the **Sysfiles** file, you can define several files to be used as **Systems** files. See the description of the **Sysfiles** file later in this chapter for details. Each entry in the **Systems** file has the following format:

```
System-name Time Type Class Phone Login
```

Each of these fields is defined in the following section.

`System-name`
This field contains the host name of the remote host.

`Time` This field is a string that indicates the day-of-week and time-of-day when the remote host can be called. The format of the *`Time`* field is:

`daytime[;retry]`

The daytime portion may be a list containing some of the following:

`Su Mo Tu We Th Fr Sa`
for individual days.

`Wk` for any weekday (Monday through Friday).

`Any` for any day.

`Never` for a passive arrangement with the remote host. This means that your host will never initiate a call to the remote host. Any call must be initiated by the remote host. In other words, your host is in a passive mode with respect to the remote host (see discussion of **Permissions** file).

Here is an example:

```
Wk1700-0800,Sa,Su
```

This example allows calls from 5:00 p.m. to 8:00 a.m., Monday through Friday, and calls any time Saturday and Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The `time` portion should be a range of times such as **0800–1230**. If no `time` portion is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, **0800–0600** means all times are allowed other than times between 6 a.m. and 8 a.m. Optionally, you may include a `retry` value to specify the minimum time (in minutes) to wait before retrying following a failed attempt. The default `retry` value is 60 minutes. If you include a `retry` value, precede it with a semicolon (**;**). For example, **Any;9** is interpreted as call at any time, but wait at least 9 minutes before retrying after a failure occurs.

Type This field contains the device type that should be used to establish the communication link to the remote host. The keyword used in this field is matched against the first field of **Devices** file entries as shown below:

```
Systems: eagle Any ACU,g D1200 3251 ogin: nuucp sword: Oakleaf2
Devices: ACU tty11 - D1200 penril
```

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol **g** to the device type **ACU**. See the information under "Protocols" in the description of the **Devices** file for details.

Class This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, **C1200**, **D1200**) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *Class* field). Some devices can be used at any speed, so the keyword **Any** may be used. This field must match the *Class* field in the associated **Devices** file entry as shown below:

```
Systems: eagle Any ACU D1200 NY3251 ogin: nuucp sword: Oakleaf2

Devices: ACU tty11 - D1200 penril
```

If information is not required for this field, use a hyphen (–) as a place holder for the field.

Phone This field is used to provide the phone number (token) of the remote host for automatic dialers or LAN switches. The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

```
Systems: eagle Any ACU D1200 NY3251 ogin: nuucp sword: Oakleaf2

Dialcodes: NY 9=1212555
```

In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (–) instructs the ACU to pause 4 seconds before dialing the next digit.

If your host is connected to a LAN switch, you may access other hosts that are connected to that switch. The **Systems** file entries for these hosts will not have a phone number in the *Phone* field. Instead, this field will contain the token that must be passed on to the switch so it will know which host your host wishes to communicate with. (This is usually just the system name.) The associated **Devices** file entry should have a **\D** at the end of the entry to ensure that this field is not translated using the **Dialcodes** file.

Login This field contains login information given as a series of fields and subfields of the format:

```
expect send
```

where *expect* is the string that is received, and *send* is the string that is sent when the *expect* string is received.

The *expect* field may be made up of subfields of the form:

```
expect[-send -expect] ...
```

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string.

For example, with **login—login**, UUCP will expect **login**. If UUCP gets **login**, it will go on to the next field. If it does not get **login**, it will send a null string followed by a New Line, then look for **login** again. If no characters are initially expected from the remote host, the characters **""** (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a New Line unless the *send* string is terminated with a **\c**.

When assembling a send/expect sequence, it is good practice not to specify the first letter of the **login:** or **password:** strings that you want UUCP to expect. The reason for this is that systems in general are inconsistent as regards the case of these first letters—some systems prompt with **login:** while others with **Login:**. Even on a given system, **login:** may appear all lowercase while **Password:** appears with a capital **P**. To avoid problems with capitalization, it is best to specify shortened forms such as **ogin:** and **sword:**.

Here is an example of a **Systems** file entry that uses an *expect-send* string:

```
wl Any ACU 1200 5556013 "" \r ogin:-BREAK-ogin: uucpx word: xyz
```

This example says expect nothing, but send a carriage return and wait for **ogin:** (for **Login:**). If you don't get **ogin:**, send a **BREAK**. If you next receive **ogin:**, send the login name **uucpx**. When you receive **word:** (for **Password:**), send the password **xyz.**

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. Table 12–2 shows the escape characters that are useful in UUCP communications.

Table 12–2 Escape Characters for UUCP Communications

Escape Character	Description
\b	Send or expect a backspace character.
\c	If at the end of a string, suppress the New Line that is normally sent. Ignored otherwise.
\d	Delay two seconds before sending or reading more characters.
\e	Echo check off.
\n	Send a New Line character.
\p	Pause for approximately 1/4 to 1/2 second.
\r	Send or expect a carriage return.
\s	Send or expect a space character.
\t	Send or expect a tab character.
\E	Start echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.)
\K	Send or expect a break character.
\N	Send or expect a null character (ASCII NUL).
\\	Send or expect a \ character.
\BREAK	Send or expect a break character.
\EOT	Send or expect EOT New Line twice.
\ddd	Collapse the octal digits (*ddd*) into a single character.

Dialcodes File

The **Dialcodes** file (**/etc/uucp/Dialcodes**) contains the dialcode abbreviations that can be used in the *Phone* field of the **Systems** file. Each entry has the format:

```
abb dial-seq
```

where *abb* is the abbreviation used in the **Systems** file *Phone* field and *dial-seq* is the dial sequence that is passed to the dialer when that particular **Systems** file entry is accessed.

The entry

```
jt 9=847-
```

would be set up to work with a *Phone* field in the **Systems** file such as `jt7867`. When the entry containing `jt7867` is encountered, the sequence 9=847–7867 would be sent to the dialer if the token in the dialer-token-pair is `\T`.

Permissions File

The **Permissions** file (**/etc/uucp/Permissions**) specifies the permissions that remote hosts have with respect to login, file access, and command execution. There are options that restrict the remote host's ability to request files and its ability to receive files queued by the local site. Another option is available that specifies the commands that a remote site can execute on the local host. Note that the **Permissions** prototype file sent with this software release is most restrictive.

Permissions File Entries
Each entry is a logical line with physical lines terminated by a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair in the following format:

```
name=value
```

Note that no white space is allowed within an option assignment.

Comment lines begin with a # and they occupy the entire line up to a New Line character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** file entries:

LOGNAME	Specifies the permissions that take effect when a remote host logs in on (calls) your host.
MACHINE	Specifies permissions that take effect when your host logs in on (calls) a remote host.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option. See your **/etc/uucp/Permissions.proto** file for more information on entry format.

Considerations
The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote hosts:

- Each login ID used by remote hosts to login for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry will have the following default permissions/restrictions: local send and receive requests will be executed, the remote host can send files to your host's **/var/spool/uucppublic** directory, and the commands sent by the remote host for execution on your host must be one of the default commands; usually **rmail**.

Options
This section describes each option, specifies how each is used, and lists the default values.

REQUEST

When a remote host calls your host and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote host can request to set up file transfers from your host. The string

```
REQUEST=yes
```

specifies that the remote host can request to transfer files from your host. The string

```
REQUEST=no
```

specifies that the remote host cannot request to receive files from your host. This is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry. A note on security: When a remote machine calls you, unless you have a unique login and password for that machine you don't know if the machine is who it says it is.

SENDFILES

When a remote host calls your host and completes its work, it may attempt to take work your host has queued for it. The SENDFILES option specifies whether your host can send the work queued for the remote host.

The string

```
SENDFILES=yes
```

specifies that your host may send the work that is queued for the remote host as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if your host is in a passive mode with respect to the remote host.

The string

```
SENDFILES=call
```

specifies that files queued in your host will be sent only when your host calls the remote host. The call value is the default for the SENDFILES option. This option is only significant in LOGNAME entries since MACHINE entries apply when calls are made out to remote hosts. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that the **uucico** program can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the **uucppublic** directory as shown in the following strings:

```
READ=/var/spool/uucppublic
WRITE=/var/spool/uucppublic
```

The strings

```
READ=/  WRITE=/
```

specify permission to access any file that can be accessed by a local user with **other** permissions.

The value of these entries is a colon-separated list of pathnames. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full pathname of a file coming in or going out. To grant permission to deposit files in **/usr/news** as well as the public directory, the following values would be used with the WRITE option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

If the READ and WRITE options are used, all pathnames must be specified because the pathnames are not added to the default list. For instance, if the **/usr/news** pathname was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote hosts to be able to write over your **/etc/passwd** file so **/etc** shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

```
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic
```

would permit reading any file except those in the **/etc** directory (and its subdirectories—remember, these are prefixes) and writing only to the default **/var/spool/uucppublic** directory. NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. There are two examples of when you would use CALLBACK. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that will be billed for the longer call.

The string

```
CALLBACK=yes
```

specifies that your host must call the remote host back before any file transfers will take place.

The default for the CALLBACK option is

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a transmission conversation will never get started.

COMMANDS

The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote host. Files and a command are sent to the target host for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote host can execute on your host. Note that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries define command permissions whether we call the remote system or it calls us.

The string

```
COMMANDS=rmail
```

indicates the default commands that a remote host can execute on your host. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:mail:lp
```

overrides the COMMAND default so that the hosts owl, raven, hawk, and dove can now execute **rmail**, **mail**, and **lp** on your host.

In addition to the names as specified above, there can be full pathnames of commands. For example,

```
COMMANDS=rmail:/usr/local/mail:/usr/bin/lp
```

specifies that command **rmail** uses the default path. The default paths for your host are **/bin**, **/usr/bin**, and **/usr/local**. When the remote host specifies **mail** or **/usr/bin/mail** for the command to be executed, **/usr/local/mail** will be executed regardless of the default path. Likewise, **/usr/bin/lp** is the **lp** command that will be executed.

Including the ALL value in the list means that any command from the remote hosts specified in the entry will be executed.

CAUTION: *If you use this value, you give the remote host full access to your host. BE CAREFUL. This allows far more access than even normal users have.*

The string

```
COMMANDS=/usr/local/mail:ALL:/usr/bin/lp
```

illustrates two points: The ALL value can appear anywhere in the string, and the pathnames specified for **mail** and **lp** will be used (instead of the default) if the requested command does not contain the full pathnames for **mail** or **lp**.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands such as **cat** and **uucp** are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE

The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your host's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged hosts have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure. (VALIDATE is merely an added level of security on top of the COMMANDS option, though it is a more secure way to open command access than ALL.)

Careful consideration should be given to providing a remote host with a privileged login and password for UUCP transactions. Giving a remote host a special login and password with file access and remote execution capability is like giving anyone on that host a normal login and password on your host.

The LOGNAME entry

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote hosts that claims to be **eagle**, **owl**, or **hawk** logs in on your host, it must have used the login **uucpfriend**. As can be seen, if an outsider gets the **uucpfriend** login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? The entry links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote host is logged in. In fact, it is an asynchronous process with no knowledge of what host sent the execution request. Therefore, the real question is how does your host know where the execution files came from?

Each remote host has its own spool directory on your host. These spool directories have write permission given only to the UUCP programs. The execution files from the remote host are put in its spool directory after being transferred to your host. When the **uuxqt** daemon runs, it can use the spool directory name to find the MACHINE entry in the **Permissions** file and get the COMMANDS list, or if the host name does not appear in the **Permissions** file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/mail \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The value in the COMMANDS option means that remote mail and **/usr/local/mail** can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the hosts listed, you are really calling either **eagle**, **owl**, or **hawk**. Therefore, any files put into one of the **eagle**, **owl**, or **hawk** spool directories is put there by one of those hosts. If a remote host logs in and says that it is one of these three hosts, its execution files will also be put in the privileged spool directory. You therefore have to validate that the host has the privileged login **uucpz**.

You may want to specify different option values for the hosts your host calls that are not mentioned in specific MACHINE entries. This may occur when there are many hosts calling in, and the command set changes from time to time. The name `OTHER` for the host name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:mail:/usr/local/Photo:/usr/local/xp
```

All other options available for the MACHINE entry may also be set for the hosts that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Poll File

The **Poll** file (**/etc/uucp/Poll**) contains information for polling remote hosts. Each entry in the **Poll** file contains the name of a remote host to call, followed by a tab character (a space won't work), and the hours the host should be called. The format of entries in the **Poll** file are:

```
sys-name hour ...
```

For example the entry:

```
eagle  0 4 8 12 16 20
```

will provide polling of host **eagle** every four hours.

The **uudemon.poll** script does not actually perform the poll. It merely sets up a polling work file (always named **C.*file***), in the spool directory that will be seen by the scheduler, which is started by **uudemon.hour**.

Sysfiles File

The **/etc/uucp/Sysfiles** file lets you assign different files to be used by **uucp** and **cu** commands as **Systems**, **Devices**, and **Dialers** files. Here are some cases where this optional file may be useful.

- You may want different **Systems** files so requests for login services can be made to different addresses than UUCP services.
- You may want different **Dialers** files to use different handshaking for **cu** and **uucp**.
- You may want to have multiple **Systems**, **Dialers**, and **Devices** files. The **Systems** file in particular may become large, making it more convenient to split it into several smaller files.

The format of the **Sysfiles** file is:

```
service=w systems=x:x dialers=y:y devices=z:z
```

where:

w is replaced by **uucico**, **cu**, or both separated by a colon.

x is one or more files to be used as the **Systems** file, with each file name separated by a colon and read in the order presented.

y is one or more files to be used as the **Dialers** file.

z is one or more files to be used as the **Devices** file.

Each file is assumed to be relative to the **/usr/lib/uucp** directory, unless a full path is given (note that the configuration files that you can change, such as **Devices**, **Systems**, and so on, are actually located in **/etc/uucp**; links in **/usr/lib/uucp** point to them). A backslash-carriage return (\<New Line>) can be used to continue an entry on to the next line.

Here's an example of using a local **Systems** file in addition to the usual **Systems** file:

```
service=uucico:cu  systems=Systems:Local_Systems
```

If this is in **/etc/uucp/Sysfiles**, then both **uucico** and **cu** will first look in **/etc/uucp/Systems**.

When different **Systems** files are defined for **uucico** and **cu** services, your machine will store two different lists of systems. You can print the **uucico** list using the **uuname** command or the **cu** list using the **uuname -c** command.

Maxuuxqts

The **Maxuuxqts** file defines the maximum number of **uuxqt** programs that can run at once. You are limited only by the number of processes you want running on your CPU. The default is 2.

Maxuuscheds

The **Maxuuscheds** file defines the maximum number of **uusched** programs that can run at once. You are limited only by the number of processes you want running on your CPU. The default is 2.

remote.unknown

The **remote.unknown** file is a shell script that executes when a machine that is not in the **Systems** file attempts to start a conversation. It will log the conversation attempt into the file **/var/spool/uucp/.Admin/foreign** and fail to make a connection. If you change the permissions of this file so it cannot execute (**chmod 000 remote.unknown**), your system will accept any conversation requests.

UUCP Spool Files

The files described in this section are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM Temporary data file. These data files are created by UUCP processes under the spool directory (i.e., **/var/spool/uucp/***X*) when a file is received from another host. The directory *X* has the same name as the remote host that is sending the file. The names of the temporary data files have the format:

`TM.`*`pid.ddd`*

where *`pid`* is a process-ID and *`ddd`* is a three-digit sequence number starting at 000.

When the entire file is received, the `TM.`*`pid.ddd`* file is moved to the pathname specified in the **C.***sysnxxxx* file (discussed later in this section) that caused the transmission. If processing is abnormally terminated, the `TM.`*`pid.ddd`* file may remain in the *X* directory. These files will be automatically removed by **uucleanup**.

LCK Lock file. Lock files are created in the **/var/spool/locks** directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The names of lock files have the format:

`LCK..`*`str`*

where *`str`* is either a device or host name.

These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on system crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

C.*name*

Work file. Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote host.

The names of work files have the format:

`C.`*`sysnxxxx`*

where *`sys`* is the name of the remote host, *n* is the ASCII character representing the grade (priority) of the work; the **uucico** code sets this priority and you may change it with **uucp**(1) and **uux**(1). *`xxxx`* is the four digit job sequence number assigned by **uucp**.

Work files contain the following information:

- Full pathname of the file to be sent or requested.
- Full pathname of the destination or user file name.
- User login name.
- List of options.
- Name of associated data file in the spool directory. If the **uucp –c** or **uuto –p** option was specified, a dummy name (**D.0**) is used.
- Mode bits of the source file.
- Remote user's login name to be notified upon completion of the transfer

D.*name*

Data file. Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

`D.`*`systmxxxxyyy`*

where *systm* is the first five characters in the name of the remote host, and *xxxx* is a four-digit job sequence number assigned by **uucp**. The four digit job sequence number may be followed by a sub-sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X.*name*

Execute file. Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

`X.`*`sysnxxxx`*

where *sys* is the name of the remote host, *n* is the character representing the grade (priority) of the work, and *xxxx* is a four digit sequence number assigned by **uucp**.

Execute files contain the following information:

- Requester's login and host name.
- Name of files required for execution.
- Input to be used as the standard input to the command string.
- System and file name to receive the command's standard output.
- Command string.
- Option lines for return status requests.

Log Files

Log files are created for each remote host with which your host communicates. There are directories for each of the **uucico**, **uucp**, **uux** and **uuxqt** commands with subdirectories under these for each machine making requests. The log files are kept in the directory **/var/spool/uucp/.Log**.

The information from the individual log files for each machine and each program (e.g., machine **dumbo** has a log file for **uucico** requests and a log file for **uuxqt** execution requests) can be accessed with the **uulog** program. These files are combined and stored in directory **/var/spool/uucp/.Old** whenever **uudemon.cleanup** is executed. This shell script saves files that are three days old. The three days can be easily modified in the **uudemon.cleanup** shell. If space is a problem, you might consider reducing the number of days the files are kept.

UUCP File Cleanup

Invoke the **uustat** program regularly to display the status of connections to various machines and the size and age of the queued requests. Use **cron** to start the **uudemon.admin** shell at least once per day to send you the current status. Of particular interest are the the age (in days) of the oldest request in each queue, the number of times a failure has occurred when attempting to reach that machine, the reason for the failure, and the age of the oldest execution request (**X.file**).

Execution files older than a few days can probably be deleted since the only reason they have not been executed is because data files required for execution were not sent. These files are usually sent at the same time as the **X.file**, so the problem is likely at the other end.

The **uucleanup** program, which is run from **uudemon.cleanup**, removes these files. Options to **uucleanup** specify the age for sending a warning message to the requester and age for deleting various files. Before deleting, the program tries to figure out what the job was and, if possible, tries to send it to the receiver. If this is not possible, it is returned to the sender.

Public Area Cleanup

To keep the local file system from overflowing when files are sent to the public area, the **uudemon.cleanup** procedure is set up with a **find** command to remove any files that are older than seven days and directories that are empty. The interval may need to be shortened if there is not sufficient space to devote to the public area.

End of Chapter

Licensed Material – Property of copyright holder(s)

Chapter 13
Login Account Management

This chapter is for systems supporting one or more login accounts. As a system manager, you will set up and manage the everyday working environment in which users function. This task includes adding and removing login accounts, creating new aliases and groups, answering users' questions, and helping users with system problems. If your system is part of a network that uses the Network Information Service (NIS) facility, formerly called the Yellow Pages (YP), some of these tasks will be the responsibility of the master server administrator. For more information on the NIS facility, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™System.*

Login Account Terms

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

login name	A valid name for logging on to the system. Also known as a *username*. A login name can be up to 32 alphabetic or numeric characters, but if you want to be compatible with traditional systems, the login name should be no more than eight characters long.
password	A unique string of alphabetic or numeric characters that allows a user access to the system. A password must be at least six characters, and a maximum of eight characters. At least one character *must* be a numeral or a special character. If you attempt to the leave the password field open when adding a user with the Add operation, the operation supplies a password that it generates itself, effectively making the login account inaccessible.
password aging	A system that forces users to change their passwords periodically. An "aged" password is one that must be changed within the specified number of weeks. When this time period lapses, the password will no longer gain entry to the system. The user must choose a new password. System administrators decide if they want to use password aging or not. See "Expert User Management Information" for details.
group	A set of users with common access privileges to a common set of files based on group ID numbers. Also known as a *group name.* People that need access to the same files can be listed in a group. For example, anyone in group **prog** could access those files associated with that group name, assuming the group permissions were set correctly for the files. Users not in the group **prog** do not have the same access rights to the files as do members of **prog**.
alias	A mailing list. An alias contains one or more user login names or aliases. If you address mail to an alias, the mail is delivered to all users listed in that alias. An alias entry consists of an alias name and a list of alias members.
user ID	A unique number that identifies a user to the system. The user ID number (**uid**) is between 100 and 60,000. The number must not include a comma. Superuser (**sysadm** and **root**) use 0. System ID numbers are 1 to 99.

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 point of the user's directory tree. The home directory is where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory. For example, user **poulet**'s home directory might be **/mach_2/poulet**.
initial program	The program invoked at the time the user logs in. Choices are:

/sbin/sh	The Bourne shell. See **sh**(1).
/usr/bin/csh	The C shell. See **csh**(1).
/usr/bin/ksh	The Korn shell. See **ksh**(1).

You may specify any executable that you or the user prefers.

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 the master system. If your system has the Network File System (ONC/NFS), and the current system is the NIS master server, you will be asked to choose local or global options in queries for user management in this chapter.

About Login Accounts

The operation for adding user login accounts has preset defaults that may be adequate for your environment. You can, if you wish, tailor these defaults using the operation User –> Login Account –> Defaults –> Set.

The following list shows how defaults are originally set on the DG/UX system:

User Id	Set to highest unassigned valid number plus one, as long as it is less than 60,000; 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**.

When you are ready, read "The User's Environment" later in this chapter. In it, we present information on global and local user profiles, environment variables, file creation permissions, default and restricted shells, an electronic bulletin board, and system mail. In addition, we offer suggestions for tracking user problems and provide a sample Trouble Log for users to fill out.

Using Groups and Aliases

Let's say that users on your system are divided into two categories: programmers and data entry people. Initially, you can assign everybody to the default group that comes with the DG/UX system. Members of group **general** are allowed access to all directories and files owned by **general**. This is a shared ownership. Later, you can assign users to additional groups. For instance, you might put programmers in group **prog** and data entry people in group **pool**. Note that group **general** is simply provided as a default to speed up the process of adding users. You can rename it or delete it. Later, you can create aliases and groups based on tasks, projects, or whatever you choose.

Aliases are simply mailing lists used by the **mailx**(1) command. The default is **everybody**. See "Adding Mail Aliases," later in this chapter, for details.

Specifying a Parent Directory and Initial Program

We do not provide a default parent directory because we can't be sure of how you've laid out your logical disks and file systems when you installed the DG/UX system. You will have to indicate the name of the directory that you wish to make the default parent directory for users.

If you have or plan to have ONC/NFS in the future, you should make sure that parent directories have unique names across systems. One way to do this is by using the host name as the parent directory name. This practice makes it easy to identify the origin and ownership of file systems in your network.

The default initial program is traditionally specified as **/sbin/sh**, but it can be any executable local shell program.

Login Account Procedures

The User menu of **sysadm** provides menus whose operations let you manage user login accounts and user login account defaults, user groups, and user mail aliases.

The following sections discuss the menus and operations in more detail.

Managing Login Accounts

A 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 in the file **/etc/passwd**. Each line in the file represents a user's login account.

The Login Account menu provides operations for adding, deleting, modifying, and listing login accounts. There is also a submenu for managing the default values that appear in the Add operation when you add a login account.

Adding Login Accounts

Select the Add operation to add a user login account. The operation adds the entry to your **/etc/passwd** file unless your system is the master server in an NIS domain, in which case you have the option of adding the entry either to the NIS **yppasswd** database or to your local **passwd** file.

For each login account you add, the operation queries you with the following prompts:

Login Name
A login name may be up to 32 characters, but to be compatible with more traditional systems, the login name should be no more than eight characters. The name must be unique on your system. If your system is in an NIS domain, the name should be unique within the domain.

Add password aging for the login account
Select this feature to set minimum and maximum age limits on the login password. Password aging is a security feature that lets you control how long a password may be in effect before the user is forced to change it. If you select password aging, the operation presents two additional queries about password aging at the end of the operation. The queries, discussed later, are **Minimum number of weeks before** *user* **may change his password** and **Maximum number of weeks until** *user* **must change his password.**

User Id
The user ID, or UID, is a number between 0 and 60,000. All numbers below 100 are reserved for the system accounts. The UID should be unique on your system. If your system is in an NIS domain, the UID should be unique within the domain. Other than serving as a unique identifier, the UID is arbitrary. The default is the highest assigned UID plus one, as long as the number is 60,000 or less; otherwise, the default is the lowest unassigned UID.

Group The user group must already exist. Add a user group with the operation User –> Group –> Add.

Home Directory
This is the directory that will be the user's home directory and current directory upon logging into the system. If the directory does not already exist, you may direct the operation to create it in a query that appears later in the operation.

User Comment
This comment may be any text that you choose to enter to describe the user or the login account.

Shell Program
Select a shell for the user. The choice is largely arbitrary and depends on the tastes of the user. For a comparison of the shells, see *Using the DG/UX™ System.*

Create home directory for *username*
If the directory you specified in the **Home Directory** prompt does not already exist, select this option to create it.

Set an initial password for *username*
Select this option if you want to assign a password for the user at this time. If you do not assign a password, the operation will assign an impossible password so that no one can log into the system through the account. To change the password for such an account after creating it with the Add operation, use the **passwd**(1) command (if the account is in the local **passwd** database) or the **yppasswd**(1) command (if the account is in the NIS **yppasswd** database). It is good practice to have the user change the password as soon as possible.

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. 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 zero. Set the expiration date in the following query.

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 his password. If the user does not change his password by this expiration period, the user will be prompted for a new password the next time he logs into the system.

If both the minimum number of weeks, set in the previous query, and the maximum number of weeks are equal to zero, the user must change his password the first time he logs into the system. Setting the minimum and maximum both to zero 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.

After you have answered 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.

Deleting Login Accounts

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**(1) 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.

Modifying Login Accounts

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.

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 may 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 "Adding Login Accounts" for more information on **passwd** entry format.

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 defaults do not suit your needs, you can change them with the operation User –> Login Account –> Defaults –> Set.

You can set and list defaults for the following login account parameters.

Group This is the user group to which the new user will belong. A user may belong to multiple groups (changing groups as desired with the **newgrp**(1) command), but you may supply only one user group name or ID number as the default group. The user group must already exist. Operations for managing user groups are in the Group menu.

Base Directory
This is the directory that will contain the user's home directory. By default, the base directory is **/home**, so a new user's home directory is **/home/***username*.

Skeleton Directory
This directory contains sample or prototype configuration files, such as **.profile**, **.login**, **.cshrc**, and so on. If you wish, you may create your own skeleton directory. 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. The files in the skeleton directory provide a foundation from which the user may begin to customize his or her own operating environment.

Shell Program
The choice of shell program typically depends on personal preference. The three choices are:

sh(1)	The Bourne shell.
csh(1)	The C shell.
ksh(1)	The Korn shell.

For a comparison of the shells, see *Using the DG/UX™ System.* As an alternative, you may specify another initial program.

Managing User Groups

Including users in *groups* allows you to grant them certain file and directory privileges while excluding users outside of the group. Use the **chmod**(1)and **chgrp**(1) commands to control group privileges in the file system.

A user may belong to multiple groups, 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**(1) command.

Local user group information is in the file **/etc/group**. Each line in the file represents a user group and gives the user group name, the group ID number, and names of members in the group.

The Group menu provides operations for adding, deleting, modifying, and listing user groups.

Adding User Groups

Select the Add operation to add an entry for a new user group. The operation 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.

A **group** entry consists of several fields, described below:

Group Name
The name may be up to 32 alphanumeric characters long, starting with a letter. To remain compatible with more traditional systems, however, you may want to limit group names to eight characters. The group name must not already exist, either in your local **group** file or, where applicable, in the global NIS database.

Group ID
The group ID, or GID, is a number between 0 and 60,000, inclusive. The numbers below 100 are reserved for system use. The GID must not already be assigned to a group name, either in your local **group** file or, where applicable, in the global NIS database. The default GID is the highest assigned GID plus one, as long as it is less than 60,000; if no such number exists, the default is the lowest unassigned number.

Group Members
List the login names of the users who will be members of the group. Separate them with commas. The login names must already exist as entries in your local **passwd** file or, where applicable, in 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.

Deleting User Groups

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**(1) 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.

Modifying User Groups

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.

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 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 Group Accounts" for more information on **group** entry format.

Managing Mail Aliases

A mail alias is a name that stands for one or more login names or mail aliases. The **sendmail**(1C)-based mail facility of the DG/UX system uses aliases when resolving mail addresses.

Mail addresses are useful for two primary purposes:

- A mail alias that resolves to a list of 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 resolves to 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.

The Mail Alias menu of the User menu provides operations for adding, deleting, modifying, and listing 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.

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.

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.

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.

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 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
```

The User's Environment

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 more information on **motd**.
- Commands to print notification of new mail messages.

Types of Profile

Profiles are of two types: global and local.

The Global Profile

The global profile is an ASCII text file, **/etc/profile** for **sh** 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.

Figure 13–1 shows the global **profile** file shipped with DG/UX.

```
#!/bin/sh
#
# Copyright (C) Data General Corporation, 1984 - 1988
# All Rights Reserved.
# Licensed Material-Property of Data General Corporation.
# This software is made available solely pursuant to the
# terms of a DGC license agreement which governs its use.
#
# Rcsid = "$What: <@(#) profile.proto.sh,v      4.1.1.3> $"
#
#       <PassStamp:_@(#)DG/UX_5.4.1__c.1.0-5.0>
#
#       The profile that all logins get before using their own .profile.

trap ""  2 3

#       Set LOGNAME
export LOGNAME

#       Set TZ.
if [ -f /etc/TIMEZONE ]
then
       /etc/TIMEZONE
elif [ -f /etc/TIMEZONE.proto ]
then
       /etc/TIMEZONE.proto
fi

#       Set TERM.
if      [ -z "$TERM" ]
then
        TERM=vt100         # for standard async terminal/console
        export TERM
fi

#       Login and -su shells get /etc/profile services.
#       -rsh is given its environment in its .profile.
case "$0" in
-su )
        export PATH
        ;;
-sh )
        export PATH

         #       Allow the user to break the Message-Of-The-Day only.
        trap "trap " 2"  2
        if [ -f /usr/bin/cat ] ; then
                cat -s /etc/motd
        fi
        trap "" 2

        if [ -f /usr/bin/mail ] ; then
                if mail -e ; then
                        echo "you have mail"
                fi
        fi
        ;;
esac

#        set the umask for more secure operation
#umask 022
trap  2 3
```

Figure 13–1 /etc/profile: Global Profile for sh and ksh Users

Figure 13–2 shows the global **login.csh** file shipped with DG/UX.

```
#
# Copyright (C) Data General Corporation, 1984 - 1988
# All Rights Reserved.
# Licensed Material-Property of Data General Corporation.
# This software is made available solely pursuant to the
# terms of a DGC license agreement which governs its use.
#
# Rcsid = "$What: <@(#) login.csh.proto.csh,v   4.1.1.2> $"
#
# /etc/login.csh
#       The .login that all users get before using their own .login.

#       Set LOGNAME
setenv USER   $LOGNAME

#       Set TZ.
if ( -r /etc/TIMEZONE.csh ) then
        source /etc/TIMEZONE.csh
endif

#       If term not set or null, set it to vt100.
if ( ! $?term ) then
    set term
endif
if ("$term" == "" ) then
        set term=vt100
        setenv TERM $term
endif

#       Protect us from ourselves.
set noclobber ignoreeof.
```

Figure 13–2 /etc/login.csh: Global Profile for csh Users

The Local Profile

The local or individual profile is **.profile** for **sh** users, and **.login** for **csh** users. These files reside in a user's home directory. These local profiles are copies of two prototype files: **/etc/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**(2) when a process begins. Since **login** is a process, the array of environment strings is made available to it. The local profiles in Figure 13–3 show how environment variables can be defined for users running the Bourne shell (**sh**(1)) or the C shell (**csh**(1)).

```
#LOCAL .PROFILE (sh 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 13–3 Local Profiles

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, *Using the DG/UX™ System* or the **sh**(1) or **csh**(1) manual pages.

Default Permissions Mode: umask

A system default controls the permissions mode of any files or directories created by a user. The DG/UX system gives default values of 666 for files and 777 for directories (see **chmod**(1M)). 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**(1) 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**(1) manual page for more information on permissions.

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.

Expert User Management Information

The remainder of this chapter provides additional information on user services. The **sysadm** program should take care of any questions you have while you are performing the major procedures for user services. Still, you should feel free to read and use this additional information to enhance your understanding and performance.

For more information on setting up terminals and keyboards to accommodate your environment, see *Customizing the DG/UX™ System.*

User Passwords

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`
: A valid name for logging onto the system. A login name can be up to 32 alphabetic or numeric characters; to remain compatible with traditional systems, you may choose to limit login names to eight characters. It is usually chosen by the user.

password: `Rm27oQak1`
: A user chooses a password and registers it with the system with the **passwd**(1) command. The encrypted form of the password appears in this field. No one but the user ever knows the real password. The actual password can be a maximum of eight characters. At least one character *must* be a numeric character or special character. This policy discourages users from choosing ordinary words as passwords. When you add a user to the file, you may use a default password, such as **passwd9**, and instruct the user to change it at the first login. Following the encrypted password, separated by a comma, there may be a field that controls password aging. For information on password aging see **passwd**(4) and **passwd**(1).

user ID: `103`
: The user ID number (**uid**) is between 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. May contain user's name, office, office phone number, home phone number, and so on. 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**(1) 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).

As noted above, the password field may contain a subfield that controls the aging of passwords. A description of how the process works can be found in **passwd**(4). The effect is to force users periodically to select a new password. If password aging is not used, a person can keep the same password indefinitely.

Password Aging

The password aging mechanism forces users to change their passwords periodically. It also prevents them from changing a password before a specified time interval. You can select 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 13–1 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 13–1 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.

Group IDs

Group IDs are a means of establishing another level of ownership of and access to files and directories. Users with some community of interest can be identified as members of the same group. Any file created by a member of the group carries the group ID as a secondary identification. By manipulating the permissions field of the file, the owner (or someone with the effective user ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the **/etc/group** file. Entries consist of the following colon–separated fields:

```
group_name:password:gid:login_names
```

A sample entry from this file is shown and explained below:

```
prog::104:reynard,poulet
```

Each entry is one line; each line has the following fields:

group_name: `prog`
: The group name can be up to 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 60,000. The number may not include any punctuation. Numbers below 100 are reserved.

login_names: `reynard,poulet`
: The login names of group members are in a comma–separated list. A user may be a member of more than one group. Nothing prevents a user from having more than one login name, however, as long as each name is unique within the system.

Sample /etc/passwd Entries

Password administration can be set up in a variety of ways to meet the needs of different organizations. Some examples are discussed in the following sections. The following shows the password aging information required to establish a new password every 2 weeks (**0**) and to deny changing the new password for 1 week (**/**).

Here is a typical login/password entry in the **/etc/passwd** file for the typical user **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**.

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.

End of Chapter

Chapter 14
Accounting Management

The DG/UX accounting system is a collection of C language programs and shell procedures with which you can monitor how system resources are being used. System-use data is logged and then organized and directed into summary files and reports which you can print or display on your terminal. Among other things, these reports are useful for helping to maintain security because you can trace who logged on when and what commands were run.

The accounting system does the following:

- Records connect sessions, logins, date changes, reboots, and shutdowns.
- Records disk use and system use for each login.
- Formats accounting data into summary files and reports (daily).
- Saves summary files, generates a report, and cleans up files (monthly).

The **sysadm** utility's System menu provides a menu containing operations for performing accounting-related tasks.

The Default Accounting System

This section summarizes the tasks involved in using the accounting system. The remainder of this chapter shows you in detail how the components of the accounting system function.

Turning on Accounting

By default, accounting is turned off. To start the accounting system, perform these steps:

1. Execute the **sysadm** operation System –> Accounting –> Start. To turn accounting off, execute the operation System –> Accounting –> Stop. Both of these operations function by calling the **turnacct**(1M) command.
2. To start accounting every time the system boots to run level 2 or higher, execute System –> Parameters –> Set. Select the option to start system accounting at boot.
3. Submit the accounting system's **cron** jobs, described below. These jobs are in the prototype **root crontab** file, **/admin/crontabs/root.proto**.

The accounting system's **cron** jobs perform the following tasks when the system is at run level 2, 3, or 4:

- Executes **runacct** at 4:00 every morning to collate statistics on connects, user activity, CPU usage, fees, disk usage, and so on. Error messages go to **/var/adm/acct/nite/fd2log**.
- Execute **ckpacct** at 5 minutes after every hour to monitor the size of the **pacct** data repository file, renaming it when it reaches its size limit (by default, 1000 blocks).

- Execute **monacct** at 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.
- Execute **dodisk** at 2:00 every morning to perform disk accounting.

The **cron** jobs that perform these actions are:

```
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
```

To set up the jobs in **root.proto** to run regularly with **cron**, add them to the **root**'s **crontab** file. See "Automating Job Execution" in Chapter 2 and the **crontab**(1) manual page for more information.

Printing Default System Daily Reports

The default accounting system produces daily reports. Use the **prdaily** command to review them. The following command line displays the reports assembled for April 11:

```
# /usr/lib/acct/prdaily 0411 ⤶
```

Without a date designation, **prdaily** prints the previous day's reports. You can also direct the output to a file using a command line like this:

```
# prdaily > yesterday_acct ⤶
```

To direct the output to a line printer, use a command line like this:

```
# prdaily | lp ⤶
```

Printing Default System Monthly Reports

The default accounting system produces monthly reports. Monthly reports are in **/var/adm/acct/fiscal**. These reports are ASCII files that you can display with a common pager such as **more** or print with **lp**.

How the Accounting System Works

The accounting system may be thought of as having three major parts: a logging mechanism, a scheduling mechanism, and a processing mechanism for the data that is logged.

- The general *logging mechanism* is activated by a script called **/usr/lib/acct/startup**, which does the major logging work for all user processes running. The **startup** script is executed according to a setting in **/etc/inittab**. By default, the **startup** script runs when the system comes up to run level 2, 3, or 4. Chapter 3 discusses run levels and the **inittab** file in detail.

- The *scheduling mechanism* is the **cron**(1M) daemon. You use **cron** as a timer to start the accounting programs at regular intervals. We demonstrated how this scheduling mechanism works earlier in "The Default Accounting System." For a discussion of **cron**, see "Automating Job Execution" in Chapter 2.
- The *processing mechanism* is a set of accounting programs in **/usr/lib/acct** that you specify for **cron** to execute.

The following programs in **/usr/lib/acct** make up the *processing mechanism* we just described. These programs are divided into two categories: user level and internal.

User Level Accounting Programs

You can invoke the following programs from the shell.

turnacct	An interface to the **accton** program that turns process accounting on or off. When switched "on", **turnacct** starts the **accton** program; "off" stops the **accton** program. When off, the accounting system is disabled.
acctcom	Searches and prints process accounting files. Note that this command is in **/usr/bin**.
chargefee	Charges a specified amount against a specified user. Does this 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.
prdaily	Displays the accounting report for a single day.
wtmpfix	Checks **/var/adm/acct/nite/wtmp** as step 2 in the execution of the **runacct** program. See "Daily Accounting with Runacct" later in this chapter for details on **runacct**.
fwtmp	Fixes corrupted accounting files by manipulating content from binary to ASCII and back. See "Fixing Corrupted Files" later in this chapter.
acctcms	Produces a summary of all processes that execute commands. See **acctcms**(1M).
acctprc1	Reads input in the form described by **acct**(4) and adds login names corresponding to user IDs. Then, it writes an ASCII line for each process giving user ID, login name, prime and nonprime CPU time (measured in *ticks*), and mean memory size in memory segment units. **Acctprc1** gets login names from **/etc/passwd**. It also reads **/var/adm/acct/nite/ctmp** to distinguish among different login names that share the same user ID.
acctprc2	Reads records in the form written by **acctprc1**, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records.
prctmp	Prints the login session record file created by **acctcon1** (normally **/var/adm/acct/nite/ctmp**).

prtacct	Formats and prints total accounting files (**tacct**).

Internal Accounting Programs

The following programs are invoked through and interact with other accounting programs. They are internal to general accounting functions and therefore you should not invoke them as regular commands.

acctwtmp	Records boot times in **/etc/wtmp**.
accton	A kernel program that monitors processes, collects data, and records that data in **/var/adm/pacct**. The programs described in **acctprc**(1M) summarize this data. Use **acctcom**(1) to examine current process data.
remove	A shell procedure that cleans up the saved **pacct** and **wtmp** files left in **/var/adm/acct/sum**.
ckpacct	Checks and controls the size of **/var/adm/pacct**. When **pacct** grows larger than 1100 blocks, **turnacct** "off" turns **accton** "off", and renames the current **pacct** file to **pacct1** and creates a new **pacct**. Finally, **ckpacct** turns **accton** back on.
runacct	The main shell program for daily accounting. See "Daily Accounting with Runacct" later in this chapter for details on **runacct.**
monacct	Uses the daily data organized by **runacct** and writes it into a monthly summary. Invoke **monacct** via **cron** once each month or each accounting period. **Monacct** creates summary files in **/var/adm/acct/fiscal.** Figure 14–4 shows how **monacct** organizes this data. See **acctsh**(1M).
dodisk	Does disk accounting on the special files in **/etc/fstab**. Creates **/var/adm/dtmp**.
diskusg	Generates disk accounting information. See **diskusg**(1M).
acctcon1	Reads **/etc/wtmp** and converts login/logoff data to a sequence of records, one per login session. The output is ASCII, giving device, user ID, login name, prime connect time (seconds), nonprime connect time (seconds), session starting time, and starting date and time. See **acctcon**(1M) for complete details.
acctcon2	This program expects a sequence of login session records as input and converts them into total accounting records (**tacct**). See **acct**(4) for the format of **tacct**.
acctdisk	Reads lines from **/var/adm/dtmp** (created by **dodisk**) containing user ID, login name, and number of disk blocks, and converts them to total accounting records that are merged with other accounting records. See **acct**(1M).
acctdusg	Computes disk resource consumption (including indirect blocks) for each login. See **acct**(1M).
acctwtmp	Called by the **shutdown**(1M) command. Writes a **utmp**(4) record containing the current time and a string of characters describing a reason, which is either **accton** or **acctoff**.
acctmerg	Merges total accounting files (**tacct**). See **acctmerg**(1M).

lastlogin	Invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date each person logged in.
nulladm	Called by most accounting scripts. Creates files with mode 664 and ensures that owner and group are **adm**.
shutacct	Invoked automatically during system shut down (via **/etc/shutdown**) to turn process accounting off and append a "reason" record to **/etc/wtmp**.

The data organized by the accounting programs is manipulated, stored, and cleaned up through interactions with files in the following directories:

```
/var/adm/acct/fiscal
/var/adm/acct/nite
/var/adm/acct/sum
```

The **/var/adm** directory contains the data collection files. The **/var/adm/acct/nite** directory contains files that are reused daily by the **runacct** procedure. The **/var/adm/acct/sum** directory contains the cumulative summary files updated by **runacct**. The **/var/adm/acct/fiscal** directory contains periodic summary files created by **monacct**.

Daily Accounting with Runacct

Runacct(1M) is the main daily accounting shell procedure. It is normally started by **cron**(1M). **Runacct** works on files that contain data on connects, fees, disk usage, and accounting processes. It also prepares daily and cumulative summary files for use by the **prdaily** program or for billing purposes. **Runacct** programs produce the following files (the **nite** and **sum** directories reside in **/var/adm/acct**):

nite/lineuse	Produced by **acctcon**(1M), which reads the **wtmp** file, and produces usage statistics for each terminal line on the system. This report is useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3:1, the line is probably failing.
nite/daytacct	Yesterday's total accounting file in binary format.
sum/tacct	The accumulation of each day's **nite/daytacct** procedure; it can be used for billing. The **monacct** shell procedure restarts the file each month or fiscal period.
sum/daycms	Produced by the **acctcms** program; contains the daily command summary. The ASCII version of this file is **nite/daycms.**
sum/cms	The accumulation of each day's command summaries. Restarted by the execution of **monacct**. The ASCII version is **nite/cms.**
sum/loginlog	Produced by the **lastlogin** shell procedure. Maintains a record of the last time each login was used.
sum/rprt.*MMDD*	Each execution of **runacct**(1M) saves a copy of the output of the **prdaily** shell procedure in this file.

Runacct(1M) does not damage files if errors occur. It tries to recognize errors, provide diagnostics, and terminate processing so that it can be restarted easily. **Runacct** writes records of its progress into a file named **active** and uses files in the **/var/adm/acct/nite** directory unless otherwise noted. By default, all diagnostics from a **runacct** are directed into the **/var/adm/acct/nite/fd2log** file.

When **runacct** starts up, its run is protected by lock files which cause any multiple invocations of the program to terminate. These lock files in the **nite** directory are named **lock** and **lock1**. The **lastdate** file contains the month and day **runacct** was last invoked and prevents more than one execution per day. If **runacct** detects an error, it writes a message to **/dev/console**, sends mail to **root**, removes the locks, saves the diagnostic files, and terminates execution.

So that **runacct** can be restartable, processing is broken down into separate re-entrant states, using a case statement inside a *while* loop. Each state is one instance of the **case** statement.

When each state completes, **nite/statefile** is updated to reflect the next state. In the next loop through the **while**, **statefile** is read and the case falls through to the next state. At the CLEANUP state, **runacct** removes the locks and terminates. States are executed in the following order:

SETUP Move active accounting files into working files. The command **turnacct switch** is executed. The process accounting files, **/var/adm/pacct**?, are moved to **/var/adm/Spacct**?.*MMDD*. The **/etc/wtmp** file is moved to **/var/adm/acct/nite/owtmp**.

wtmpfix Verify the integrity of **/etc/wtmp**, correcting date changes if necessary. The **wtmp** file in the **nite** directory is checked by the **wtmpfix** program. Some date changes cause the **acctcon1** shell procedure to fail, so **wtmpfix** tries to adjust the time stamps in the **wtmp** file if a date change record appears.

CONNECT1 Produce connect session records. Connect session records are written to **ctmp** in binary form. The **lineuse** file is created, and the **reboots** file is created showing all of the boot records the **wtmp** file.

CONNECT2 Convert session records into total accounting records. **Ctmp** is converted to **ctacct**.*MMDD* which contains accounting records.

PROCESS The **acctprc1** and **acctprc2** programs convert the process accounting files, **/var/adm/Spacct**?.*MMDD*, into total accounting records in **ptacct**?.*MMDD*. The **Spacct** and **ptacct** files are correlated by number so that if **runacct**(1M) fails, **Spacct** files are not reprocessed.

NOTE: When restarting **runacct** in this state, remove the last **ptacct** file; the **ptacct** file will not be complete.

MERGE Merge the process accounting records with the connect accounting records to form the **daytacct** file.

FEES Merge any ASCII records from the file **fee** with **daytacct**.

DISK On the day after the **dodisk** procedure runs, merge **disktacct** with **daytacct**. This merge forms the daily total accounting records.

MERGETACCT
Merge **daytacct** with **sum/tacct**, the cumulative total accounting file. Each

	day, **daytacct** is saved in **sum/tacct***MMDD*, so that **sum/tacct** can be recreated if it is corrupted or lost.
CMS	Merge today's command summary with the cumulative command summary file **sum/cms**. Produce ASCII and internal format command summary files.
USEREXIT	Include installation dependent (local) accounting programs here.
CLEANUP	Clean up temporary files, run **prdaily** and save its output in **sum/rprt***MMDD*, remove the locks, then exit.

At the completion of the last state, an ASCII file exists in directory **sum**; it has a four-digit suffix on it where the first two digits represent the month, and the last two represent the day; for example, **rprt1120** is the report for November 20. To display the previous day's report, execute **/usr/lib/acct/prdaily**. For a monthly summary, you can print the reports assembled by **monacct**. These reports, which have a two-digit suffix representing the month, are in directory **fiscal**. The report for November would be **fiscrpt11**.

Runacct Accounting Reports

Runacct generates the following accounting reports:

- Daily line usage
- Daily usage by login name
- Daily total command summaries
- Monthly total command summaries
- Last login

You cannot use **runacct** to generate any of these reports individually. The **runacct** program generates all five reports at once. Sample reports and meanings of their data follow.

Daily Line Usage

The first part of the daily line usage report is the **from/to** banner. The banner displays the time the last accounting report was generated and the time the current accounting report is generated. It is followed by a log of system reboots, shutdowns, recoveries, and any other record dumped into **/etc/wtmp** by the **acctwtmp** program.

The second part of the report is a breakdown of line usage. `Total Duration` tells how long the system was accessible through the terminal lines.

Figure 14–1 is an example daily line usage report.

```
Nov 11 16:00 1992  DAILY REPORT For DG/UX page 1

from Tue Nov 10 08:27:00 1992
to   Wed Nov 11 04:00:42 1992

2    shutdown

1    runacct

Total Duration is 1174 Minutes

Line     Minutes     Percent     # Sess     # On     # Off
-------------------------------------------------------
tty01          0           0          1        1         0

tty02        506          43          1        1         1

tty03         48           4          5        5         5

console       41           3          1        1         1

TOTALS       693          --          9        9         8
-------------------------------------------------------
```

Figure 14–1 Line Usage Report

The columns in the report above are defined as follows:

`Line`	Terminal line or access port
`Minutes`	Total minutes of line use during the accounting period
`Percent`	Total minutes of line use divided into the total duration
`# Sess`	Number of times this port was accessed for a **login**(1) session
`# On`	Number of times the port was used to log a user in to the system.
`# Off`	Number of times a user logged off, and any interrupts on that line. Generally, interrupts occur on a port when 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.

During normal operation, you should monitor the size of **/etc/wtmp** because this is the file from which the connect accounting is geared. If the file grows rapidly, execute **acctcon1** to see which tty line is the noisiest. If interruption is occurring at a rapid rate, it can affect general system performance.

Daily Usage by Login Name

The daily usage by login name report gives a breakdown of system resource use for each user. Its data consists of:

UID	User ID.
LOGIN NAME	Login name of the user. (There can be more than one login name for a single user ID.)
CPU (MINS)	CPU time used, divided into PRIME and NPRIME (nonprime) See "Updating Holidays," later in the chapter.
KCORE-MINS	Total memory a process used, in kilobytes per minute. Divided into PRIME and NPRIME.
CONNECT (MINS)	How long a user was logged into the system, by PRIME and NPRIME time. If this time is long and the column #OF PROCS is low, this user rarely uses the terminal.
DISK BLOCKS	When the disk accounting programs have been run, their output is merged into the total accounting record (**tacct.h**) and shows up in this column. **Acctdusg** does disk accounting.
# OF PROCS	Number of processes invoked by the user. Large numbers may indicate that a shell procedure looped.
# OF SESS	Number of times the user logged in to the system.
# DISK SAMPLES	Number of times the disk accounting was run to obtain the average number of disk blocks.
FEE	Total accumulation of total charges against the user. You use the **chargefee**(1M) command to charge a user for special services, such as restoring files and mounting tapes.

Figure 14–2 is a sample report for each user.

```
Nov 11 04:42 1992 Daily Usage Report

Uid Login    CPU (Min)   Kcore-Mins   Connect(Min)  Disk    # Of  # Of # Disk
    Name   Prime NPrime Prime NPrime Prime NPrime  Blocks  Procs Sess Samples Fee
---------------------------------------------------------------------------------

0   TOTAL  37    235    1198  467190 56    130     0       6142  11   0       0

0   root   5     1      472   84     41    0       0       712   1    1       0

2   bin    0     0      0     0      0     0       397415  0     0    1       0

3   sys    0     0      0     0      0     0       0       0     1    0       0

4   adm    0     1      0     78     0     0       17630   291   0    1       0

5   uucp   2     2      182   249    0     0       0       740   2    1       0

8   mail   0     0      13    3      0     0       0       13    4    1       0

201 moe    5     0      718   0      0     0       0       151   1    1       0

202 larry  0     1      15    5      0     99      0       13    1    1       0

203 curly  25    5      3069  371    15    31      3345    404   2    1       0

204 poulet 0     0      0     0      0     0       32      0     0    1       0
```

Figure 14–2 Daily Usage by Login Name

Daily and Monthly Total Command Summaries

The daily and monthly total command summary reports are similar, but the Daily Command Summary reports only the current accounting period; the Monthly Total Command Summary reports from the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of **monacct**.

These reports tell which commands are used most. Since you know which system resources the commands use, you can tune the system accordingly. These reports are sorted by `TOTAL KCOREMIN` (see below), which is a good way to calculate drain on a system.

`COMMAND NAME`	The name of the command. All shell procedures are reported under the name **sh**.
`NUMBER CMDS`	Number of times a command was invoked.
`TOTAL KCOREMIN`	Total KB 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.

Licensed Material – Property of copyright holder(s)

`MEAN CPU-MIN`	Mean of the NUMBER CMDS and TOTAL CPU-MIN.
`HOG FACTOR`	Ratio of system availability to system usage: hog factor = (total CPU time) / (elapsed time) This measures the total available CPU time the process used.
`CHARS TRNSFD`	Number of characters manipulated by the **read**(2) and **write**(2) system calls. The value in this column may be negative.
`BLOCKS READ`	Total physical block reads and writes that a process performed.

Figures 14–3 and 14–4 are examples of daily and monthly command summaries.

```
Nov 11 04:42 1992 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
```

Figure 14–3 Daily Command Summary

```
Nov 11 04:42 1992 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

--------------------------------------------------------------------------------
```

Figure 14–4 Monthly Command Summary

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.

```
Nov 11 04:42 1992 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
```

Figure 14–5 Last Login Report

A field of all zeros means that login has not been used since the last invocation of the **lastlogin** program.

Recovering from Failure

If the system crashes, **/usr** runs out of space, or a **wtmp** file is corrupted, **runacct** fails. If the **active***MMDD* file exists, check it first for error messages. If the **active** file and **lock** file exist, check **fd2log** for error messages.

Runacct may produce the following error messages. We suggest ways to recover from them.

`acctg already run for date: check /var/adm/acct/nite/lastdate`

> The date in **lastdate** and today's date are the same. Remove **lastdate**.

`connect acctg failed: check /var/adm/acct/nite/log`

> The **acctcon1** program encountered a bad **wtmp** file. Use **fwtmp** to fix the bad file. See "Fixing Corrupted Files" later in this chapter.

`locks found, run aborted`

> The files **lock** and **lock1** were found in **/var/adm/acct/nite**. Remove these files before restarting **runacct**.

`Spacct?.MMDD already exists`

> File setups have already run. Check status of files, then run setups manually. See "Restarting Runacct."

`turnacct switch returned rc=?`

> Check the integrity of **turnacct** and **accton** by ensuring that the **accton** program is owned by **root** and has the setuid bit set. See **setuid**(2).

`/var/adm/acct/nite/wtmp.MMDD already exists, run setup manually.`

> See "Fixing Corrupted Files" later in this chapter.

`wtmpfix errors see /var/adm/acct/nite/wtmperror`

> **Wtmpfix** detected a corrupted **wtmp** file. Use **fwtmp** to fix the file.

Restarting Runacct

Runacct called without arguments assumes this is the first invocation of the day. You must use the argument *MMDD* if **runacct** is being restarted; *MMDD* specifies the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of **statefile**. To override **statefile**, include the desired state on the command line. As we said earlier, **runacct** is normally started by **cron**. But should you need to start **runacct** from the command line, here are three ways you might do it:

To start **runacct**, type:

```
# nohup runacct 2> /var/adm/acct/nite/fd2log & ↵
```

To restart **runacct** specifying *MMDD* (0601), type:

```
# nohup runacct 0601 2> /var/adm/acct/nite/fd2log & ↵
```

To restart **runacct** at a specific state, such as **wtmpfix**, type:

```
# nohup runacct 0601 wtmpfix 2> /var/adm/acct/nite/fd2log & ⮐
```

In the above examples, the **2>** sends the standard error output to the file named **fd2log**; check this file periodically for error messages. Specifying *MMDD* creates a new **active0601** file, dated June 1.

Fixing Corrupted Files

Sometimes, a file is corrupted or lost. Although you may restore some files from backup tapes, you must fix the **wtmp** and **tacct** files yourself.

Fixing wtmp Errors

Wtmp files record who logged in and when. When the date is changed and the DG/UX system is in multi-user mode, a set of date change records is written into **/etc/wtmp**. The **wtmpfix** program adjusts the time stamps in the **wtmp** records when a date change is encountered.

Some combinations of date changes and reboots, however, result in nonsense lines being added to **/etc/wtmp**; these lines cause **acctcon1** to fail. When this happens, **wtmp***MMDD* is created. The following procedure shows you how to fix this file. At the editing step below, you'll see binary lines mixed in with ASCII lines. Fixing this file consists of deleting the binary lines.

1. Go to directory **/var/adm/acct/nite**.
2. Enter the following command at the prompt:

   ```
   # 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. Edit **xwtmp** and delete all binary lines.
4. When you're finished editing, convert from ASCII back to binary.

   ```
   # fwtmp -ic < xwtmp > wtmp.MMDD ⮐
   ```

If the **wtmp** file is beyond repair, create a null **wtmp** file 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 at the prompt:

   ```
   # acctmerg -v < tacct.MMDD > xtacct ↵
   ```

   This command line executes the **acctmerg**(1M) program on the contents of **tacct.***MMDD*, and stores the output in file **xtacct**, 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. Type the following at the prompt.

   ```
   # acctmerg -i < xtacct > tacct.MMDD ↵
   ```

Remember that **monacct** removes all the **tacct.***MMDD* files; therefore, when you merge these files together you recreate **sum/tacct**.

Updating Holidays

The file **/usr/lib/acct/holidays** contains the prime/nonprime table for the accounting system. Edit the table to reflect your holiday schedule for the year. The table format has three types of entries:

1. *Comment Lines:* Comment lines have an asterisk in column 1.
2. *Year Designation Line:* This line should be the first data line (noncomment line) in the file; it must appear only once. It has three fields: year, prime time, and 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.

3. *Holiday Lines:* These entries follow the year designation line, and have the format:

   ```
   day-of-year  Month  Day  Description of Holiday
   ```

   The day-of-year field is a number between 1 and 366, indicating the day for the corresponding holiday; leading blanks, tabs, and spaces are ignored. The other three fields are commentary, and are not currently used by other programs.

The accounting system will not function properly unless a **holidays** file exists. You will receive a **/usr/lib/acct/holidays.proto** as part of the DG/UX system. When you boot the system for the first time, **holidays.proto** will automatically be copied to **holidays**. Add entries to **holidays** as suits your needs. A sample file follows:

```
*
* Copyright (C) Data General Corporation, 1984 - 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 08301700
*
* 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 Directories and Files

This last section of the chapter briefly describe all the data files within the DG/UX accounting structure.

Files in the /var/adm Directory

The following files reside in the **/var/adm** directory.

diskdiag	Diagnostic output during the execution of disk accounting programs.
dtmp	Output from the **acctdusg** program.
fee	Output from the **chargefee** program, ASCII **tacct** records.
pacct	Active process accounting file.
pacct*?*	Process accounting files switched via **turnacct**.
Spacct*?.MMDD*	Process accounting files for *MMDD* during execution of **runacct**.

Files in the /var/adm/acct/nite Directory

The following files reside in the **/var/adm/acct/nite** directory.

active	Used by **runacct** to record progress and print warning and error messages.
cms	ASCII total command summary used by **prdaily**.
ctact.*MMDD*	Connect accounting records in binary.
ctmp	Output of **acctconl** program; connect session records in binary.
daycms	ASCII daily command summary used by **prdaily**.
dayacct	Total accounting records for one day in binary.
disktacct	Disk accounting records in binary; they are created by **dodisk.**
fd2log	Diagnostic output during execution of **runacct** (see **cron** entry).
lastdate	Last date **runacct** executed in *MMDD* format.
lock lock1	Used to control serial use of **runacct**.
lineuse	TTY line usage report used by **prdaily**.
log	Diagnostic output from **acctconl.**
log*MMDD*	Same as log after **runacct** detects an error.
reboots	Contains beginning and ending dates from **wtmp** and a listing of reboots.
statefile	Contains current state of a **runacct** run.
tmpwtmp	**wtmp** file corrected by **wtmpfix.**
wtmperror	Place for **wtmpfix** error messages.
wtmperror*MMDD*	Same as **wtmperror** after **runacct** detects an error.
wtmp.*MMDD*	Previous day's **wtmp** file.

Files in the /var/adm/acct/sum Directory

The following files reside in the **/var/adm/acct/sum** directory.

cms	Total command summary file for current fiscal year in internal summary format.
cmsprev	Command summary file without latest update.
daycms	Command summary file for yesterday in internal summary format.
loginlog	Created by **lastlogin.**
pacct.*MMDD*	Concatenated version of all **pacct** files for *MMDD*, removed after reboot by **remove** procedure.
rprt.*MMDD*	Saved output of **prdaily** program.
tacct	Cumulative total accounting file for current fiscal year.

tacctprev	Same as **tacct** without latest update.
tacct.*MMDD*	Total accounting file for *MMDD*.
wtmp.*MMDD*	Saved copy of wtmp file for *MMDD*; removed after reboot by **remove** procedure.

Files in the /var/adm/acct/fiscal Directory

The following files reside in the **/var/adm/acct/fiscal** directory.

cms	Total command summary file for a fiscal period.
fiscrpt	Report similar to **prdaily** for a fiscal period.
tacct	Total accounting file for a fiscal period.

End of Chapter

Licensed Material – Property of copyright holder(s)

Chapter 15
Using CD–ROM, Magneto-Optical, and Diskette Drives

This chapter contains information about the installation and use of the following SCSI drives:

- CD–ROM (Compact Disk–Read–Only Memory) disk drive.
- Multiple–read/write magneto-optical disk drive.
- Diskette drive.

For information about disk drives and devices in general, see *Customizing the DG/UX System.*

General Information

The information in this section applies to all three types of drives: CD–ROM, magneto-optical, and diskette drives.

Make sure you have the proper terminator on the last device in a chain of SCSI devices; otherwise, your system will panic.

If the total length of SCSI cable for a particular adapter is greater than 19 feet, you could get errors about problems in a device's physical file table. Excessive cable length could also result in problems when trying to access the last device on the chain.

If you have multiple CD–ROM drives, each must have a unique SCSI ID. Multiple magneto-optical or diskette drives, on the other hand, may be clustered so that as many as four share the same SCSI ID. If you put multiple units on the same SCSI ID, you will need to use device specifications that include a third argument (for unit number). For example, the following specifications represent three 5.25-inch diskette drives at SCSI ID 3:

```
sd(insc(),3,0)
sd(insc(),3,1)
sd(insc(),3,2)
```

For every device on your system, there is a short name created at boot time that you can use to refer to the device. You can find a table showing the long name/short name pairs in the file **/etc/devlinktab**. For example, a workstation's SCSI tape drive has this specification in DG/UX common device specification format:

```
st(insc(),4)
```

With this name in mind, you can locate the device's entry (which has a very similar long name) in the **devlinktab** file. The example below shows the device's entry (as well as the comment lines from the file that have the table headers).

```
# directory     short     long
#
/dev/rmt        0         st(insc@7(FFF8A000),4,0)
```

From this entry, you know that the short name for **st(insc(),4)** is **/dev/rmt/0**. The device is a tape drive, so it has a no–rewind version too, which is **/dev/rmt/0n**.

You use a device's short name when you need to write to it (such as a tape) or mount it as a file system.

The precise adapter specification, SCSI id, and unit number depend on the kind of hardware you have and how the jumpers are set.

Using the CD–ROM Drive

The DG/UX system supports the following kinds of file systems on CD–ROM disks:

DG/UX This is the typical DG/UX disk file system. For system administration commands and **sysadm** operations, this is a file system of type **dg/ux**. The DG/UX system recognizes this file system by default.

MS–DOS For system administration commands and **sysadm** operations, this is a file system of type **dos**. Before using a CD–ROM disk of this type, you must configure the MS–DOS file system manager into your kernel by adding the name **dfm()** in your system file on a line by itself before building a new kernel.

High Sierra and ISO 9660
For system administration commands and **sysadm** operations, these file systems are of type **cdrom**. Before using a CD–ROM disk of this type, you must configure the High Sierra file system manager into your kernel by adding the name **hfm()** in your system file on a line by itself before building a new kernel.

To build a kernel, see Chapter 4.

After inserting a disk into a CD–ROM drive, there are two things you must do before you can use it:

1. Register the device using the **sysadm** operation Device –> Disk –> Physical –> Register.

   NOTE: If the disk in the drive is already hardware formatted according to the ISO 9660 or High Sierra standard, do not register it. Check your drive's hardware documentation to find out whether your drive follows either of these standards.

2. Mount the device's file systems using the **sysadm** operation File System –> Local Filesys –> Mount. If the disk is ISO 9660 or High Sierra, mount file systems as type **cdrom**. If the disk is an MS–DOS disk, mount file systems as type **dos**. If the disk contains DG/UX file systems, mount them as type **dg/ux**.

To take a disk out of the CD–ROM drive, first unmount any file systems (if mounted) and deregister the device (if registered).

Using the Magneto-Optical Drive

You may use a magneto-optical drive as a tape by writing to it using **tar**(1) or **cpio**(1), or you may use to contain file systems. To prepare a new magneto-optical disk to contain file systems, insert the disk into the drive and follow these steps:

1. Prepare the disk by putting system areas on it with Device –> Disk –> Physical –> Soft Format –> Create System Areas.
2. Register the disk with Device –> Disk –> Physical –> Register.
3. Create logical disks with Device –> Disk –> Logical –> Create.
4. Create a file system on each logical disk with File System –> Local Filesys –> Create.
5. Add the file systems with File System –> Local Filesys –> Add. When queried, supply a file system type of **dg/ux**.
6. Mount the file systems with File System –> Local Filesys –> Mount.

The file systems on the disk are now ready for use.

Before removing a magneto-optical disk from the drive, first unmount any file systems (if mounted) and deregister the device (if registered).

You cannot register this device unless there is a disk in the device.

You cannot use **sysadm** to create a backup on a magneto-optical disk. To make a backup on a magneto-optical disk, use **dump2**(1M). A backup to magneto-optical disk media is limited to one disk.

Using the Diskette Drive

This section contains information specific to diskette drives.

Assigning Unit Numbers

If you intend to install several diskette drives on the same SCSI ID, you need to observe these rules:

- You may have four drives total, units **0** through **3**.
- If the devices on the SCSI ID are all 5.25-inch diskette drives, they may have any of the four available unit numbers (make sure the hardware is jumpered correctly).
- If any of the devices on the SCSI ID is a 3.5-inch diskette drive, then only units 0 and 1 may be used for 3.5-inch drives, and only units 2 and 3 may be used for 5.25-inch drives. If you have only 3.5-inch drives, they are still restricted to units 0 and 1.

For example:

- If you have three 5.25-inch diskette drives, they can be units 0 through 2.
- If you decide to add a 3.5-inch drive to the same adapter with the three 5.25-inch drives, however, the 3.5-inch drive may be unit 0 or 1, and two of the 5.25-inch drives may be units 2 and 3. The third 5.25-inch drive must use a different SCSI ID.

- If you have three 3.5-inch drives, you may put two on the same SCSI ID, at units 0 and 1, but the third must have a different SCSI ID.

You need to make sure the SCSI ID, unit number, and adapter card straps are set correctly. The adapter card has the SCSI ID settings and the strap settings, and the drive itself has the unit setting.

If your peripheral housing unit contains any 3.5-inch diskette drives, set the straps on the TEAC adapter card to H, F, LEV, STL, and PAR. If the housing unit contains only 5.25-inch diskette drives, set the straps on the TEAC adapter card to G, F, LEV, STL, and PAR. The H is for 1.44 MB format, G is for 1.2 MB format, and F is for 720 KB format.

If you have two 5.25-inch diskette drives in the same peripheral housing unit, set the IS strap so the drive does not reset the ready state every time the drive changes density between the two 5.25-inch diskette drives.

If you have two 3.5-inch diskette drives in the same peripheral housing unit, set the HHI strap so that high density input is active high when a high density diskette is accessed and active low when a low density diskette is accessed.

Formatting a Diskette

Formatting a diskette is the same as creating a file system on it. Typically, you create system areas and logical disks on a physical disk before creating file systems on it. Diskettes are much smaller than typical disks, however, and system areas take up too much space. On a diskette it is more efficient not to create system areas and to create a single file system directly on the physical disk instead. To create a file system on a diskette, follow these steps:

1. Use File System –> Local Filesys –> Create to format the diskette. You need to know the pathname of the physical device, for example, **/dev/pdsk/1**. You may have to look in **/etc/devlinktab** to see how device file names map to device specifications. If your device does not appear in **devlinktab**, you need to verify that the diskette drive is installed correctly and configured in your kernel before rebooting your system. See "Building a Kernel" in Chapter 4 to configure a device in your kernel.

   By default, the Create operation makes a DG/UX file system. To create an MS-DOS file system instead, specify the `dos` (or `pc`) option and a density value to **mkfs**. For 5.25-inch diskettes, the supported density values are `360kb` and `1220kb`. For 3.50-inch diskettes, the supported density values are `720kb` and `1440kb`. See the **mkfs**(1M) manual page for complete information on **mkfs** options.

   You may also obtain pre-formatted diskettes from your Data General representative.

   To access an MS-DOS file system, the MS–DOS file manager driver, **dfm()**, must be configured in your kernel. When you build your kernel using the System –> Kernel –> Build operation, add the entry for `dfm()` when editing the system file.

2. Add the file systems with File System –> Local Filesys –> Add.

   Specify the appropriate file system type when prompted, such as `dos` or `dg/ux`. The device you mount is the same one on which you created the file system.

3. Mount the file system with File System –> Local Filesys –> Mount.

Changing Diskettes

Before removing a diskette from the drive, you should unmount the file system. If you remove the diskette without unmounting the file system, you may still unmount the file system even though the diskette is not present.

If you put another diskette in the drive without having unmounted the previous one, however, the system will prompt you for the previous diskette. Until you re–insert the previous diskette and unmount its file system, you may not access the drive.

Using a Diskette as a Tape

Instead of using the diskette as a file system, you may use it as a tape. Like a tape, you can write to the diskette using **tar**, **cpio**, **dump**, **dump2**, or **dd**. Refer to the device as **/dev/rpdsk** or **/dev/pdsk**. Before you can use the device this way, you must unmount it (if mounted) and deregister it (if registered).

Because diskettes do not have tape marks (as found on magnetic tape), you cannot use multiple diskettes with **tar**, **dump**, **dump2**, or **dd**. The **cpio** command, on the other hand, allows you to use multiple diskettes when archiving files.

You cannot use **sysadm**, that is, the File System –> Backup –> Create operation, to dump to a diskette drive. You may dump to a diskette from the shell using the **dump2**(1M) command, but you are limited to one diskette.

If you boot the system while a diskette with a **tar** or **cpio** format file is in the diskette drive, you may see an error having to do with the physical file table. Ignore this error. If the diskette drive is registered as a file system when you try to write to it as a file (with **tar** or **cpio**), you will receive the error, `Conflict on open`. Deregister the device before writing to it with **tar** or **cpio**.

End of Chapter

Appendix A
UUCP Error and Status Messages

This appendix contains error and status messages for the UUCP system.

UUCP Error and Status Messages

This section lists two types of error and status messages associated with UUCP connections.

-
- ASSERT errors are recorded in the **/var/spool/uucp/.Admin/errors** file.
-
- STATUS errors are recorded in individual machine files found in the **/var/spool/uucp/.Status** directory.

ASSERT Error Messages

When a UUCP process fails, ASSERT error messages may be generated and recorded in **/var/spool/uucp/.Admin/errors**. These messages include the file name, SCCS id, line number, and the text listed below. ASSERT messages reflect conditions which the system manager must correct. These errors are usually due to system problems.

Assert Error Message	Description/Action
`BAD LINE`	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
`BAD SPEED`	A bad line speed appears in the **Devices/Systems** files (Class field).
`BAD LOGIN_UID`	The uid cannot be found in the **/etc/passwd** file. The file system is in trouble, or the **/etc/passwd** file is inconsistent.
`BAD UID`	Same as previous.
`CAN'T ALLOCATE`	A dynamic allocation failed.
`CAN'T CHDIR`	A **chdir**() call failed.
`CAN'T CHMOD`	A **chmod**() call failed.
`CAN'T CREATE`	A **create**() call failed.
`CAN'T CLOSE`	A **close**() or **fclose**() call failed.

`CAN'T FORK`	An attempt to **fork** and **exec** failed. The current job should not be lost, but will be attempted later (**uuxqt**). No action need be taken.
`CAN'T LINK`	A **link**() call failed.
Assert Error Message	**Description/Action**
`CAN'T LOCK`	An attempt to make a **LCK** file (in **/var/spool/locks**) failed.
`CAN'T OPEN`	An **open**() or **fopen**() failed.
`CAN'T READ`	A **read**(), **fgets**(), etc., failed.
`CAN'T STAT`	A **stat**() call failed.
`CAN'T WRITE`	A **write**(), **fwrite**(), **fprint**(), etc., failed.
`CAN'T UNLINK`	An **unlink**() call failed.
`WRONG ROLE`	This is an internal logic problem. See the release notice for instructions on filling out a Software Trouble Report (STR) to send to the Data General Customer Support Center.
`FILE EXISTS`	The creation of a work file (**C.***file*) or data file (**D.***file*) was attempted, but the file exists. This occurs when there is a problem with the sequence file access. This problem usually indicates a software error. See the release notice for instructions on filling out a Software Trouble Report (STR) to send to the Data General Customer Support Center.
`ULIMIT TOO SMALL`	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
`SYSLST OVERFLOW`	An internal table in **gename.c** overflowed. A big/strange request was attempted.
`TOO MANY FILES`	Same as previous.
`RETURN FROM fixline ioctl`	An **ioctl**() call, which should never fail, failed. There is a system driver problem. See the release notice for instructions on filling out a Software Trouble Report (STR) to send to the Data General Customer Support Center.
`PERMISSIONS file: BAD OPTION`	There is a bad line or option in the **Permissions** file.
`PKCGET READ`	The remote machine probably hung up. No action need be taken.
`PKXSTART`	The remote machine aborted in a nonrecoverable way. This can generally be ignored.
`SYSTAT OPEN FAIL`	There is a problem with the modes of **/var/spool/uucp/.Status**, or there is a file with bad modes in the directory.
`TOO MANY LOCKS`	There is an internal problem. See the release notice for instructions on filling out a Software Trouble Report (STR) to send to the Data General Customer Support Center.

`XMV ERROR`	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were supposed to be checked before this process was attempted.

STATUS Messages

Status messages are stored in the **/var/spool/uucp/.Status** directory. This directory contains a separate file for each remote machine that your system attempts to communicate with. These files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common messages that may appear in these files.

Status Message	**Description/Action**
`ASSERT ERROR`	An ASSERT error occurred. Check the **/var/spool/uucp/.Admin/errors** file for the error message.
`BAD LOGIN/MACHINE COMBINATION`	The machine called us with a login/machine name that does not agree with the **Permissions** file.
`CALLBACK REQUIRED`	The called machine requires that it call your DG/UX system.
`CALLER SCRIPT FAILED`	This is usually the same as `DIAL FAILED`; however, if it occurs often, suspect the caller script in the **dialers** file. Use **Uutry** to check.
`CAN'T ACCESS DEVICE`	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
`CONVERSATION FAILED`	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
`DEVICE FAILED`	The open of the device failed.
`DEVICE LOCKED`	The calling device to be used is currently locked and in use by another process.
`DIAL FAILED`	The remote machine never answered. It could be a bad dialer or the wrong phone number.
`LOGIN FAILED`	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer–Token–Pairs script.
`NO DEVICES AVAILABLE`	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
`OK`	Things are OK.
`REMOTE DOES NOT KNOW ME`	The remote machine does not have the nodename of your computer in its **Systems** file.

`REMOTE HAS A LCK FILE FOR ME`	The remote site has a **LCK** file for your computer. They could be trying to call your machine. If they have an older version of UUCP, the process that was talking to your machine may have failed leaving the **LCK** file. Check to see if the remote process that created the **LCK** file is hung.
`REMOTE REJECT AFTER LOGIN`	The login used by your computer to login does not agree with what the remote machine was expecting.

Status Message	**Description/Action**
`REMOTE REJECT, UNKNOWN MESSAGE`	The remote machine rejected the communication with your computer for an unknown reason. The remote machine may not be running a standard version of UUCP.
`STARTUP FAILED`	Login succeeded, but initial handshake failed.
`SYSTEM NOT IN Systems`	The system is not in the **Systems** file.
`TALKING`	Local UUCP and remote UUCP are communicating successfully.
`WRONG TIME TO CALL`	A call was placed to the system at a time other than what is specified in the **Systems** file.
`WRONG MACHINE NAME`	The called machine is reporting a different name than expected.

End of Appendix

Appendix B
fsck Error Conditions

The **fsck**(1M) program checks the internal consistency of file systems, repairing inconsistencies when invoked to do so. At boot, the system runs **fsck** on all local file systems according to options specified in the **fstab** file. For more information on file systems and how to use **fsck** to check them, see Chapter 8.

When **fsck** detects an inconsistency, it reports the error condition to the operator. If a response is required, **fsck** prints a prompt message and waits for a response. This appendix explains the meaning of each error condition, possible responses, and related error conditions.

In "Error Messages for Phased Checking," the error conditions are organized by the phase of the **fsck** program in which they can occur. The error conditions that may occur in more than one phase are discussed under "General Error Messages."

Error messages that occur only during fast recovery file system checking (when checking a file system that had **fsck** logging turned on) appear "Error Messages Exclusive to Fast Recovery Checking" at the end of the appendix.

The following error messages are presented in their basic form. Fatal errors (such as during **fsck –p**) cause the error message to be prefaced by the string `Fatal Error:`. Running with **–p** also causes messages to be preceded by the name of the file system to which the message applies. The following abbreviations appear in the description of error messages:

B	A (decimal) disk block number.
N	A decimal number.
O	An octal number.
C	A character.
D, *F*	A directory name, file name or pathname string.
I	An inode description string. At the very least, this will consist of the inode number. If possible, the inode's size, file type, file mode, UID, GID, time of last modification, owner name, group name and pathname will also be present.

Error Messages for Phased Checking

The **fsck** program checks file systems in phases only if **fsck** logging was not turned on for the file system. This section lists errors you may see during the typical, phased **fsck** check. Some of these error messages may also appear during checking of a file system for which **fsck** logging was turned on.

The section at the end of this appendix lists errors that can appear only during checking of file systems for which **fsck** logging was turned on.

General Error Messages

The messages described in this section may appear at any time during an **fsck** session.

Cannot allocate memory for internal tables (N bytes requested)

The **fsck** program cannot allocate enough memory; this can only occur during stand–alone **fsck**. The **fsck** program will abort. Bring up your system and use the **fsck** command instead.

Cannot read block B

A disk read of block number *B* has failed. The **fsck** program treats the block it could not read as if it were filled with all zeroes, and continues execution, but the file system is not marked as mountable upon conclusion of checking. Use **diskman** to remap the bad block *B* and run **fsck** again.

Cannot write block B

A disk write of block number B has failed. The **fsck** program continues execution, but the file system being checked is not marked as mountable upon conclusion of checking. Use **diskman** to remap the bad block *B*, and run **fsck** again.

Fork failed

The **fsck** program has failed in an attempt to spawn a child process. This will only occur when running **fsck** with the **–p** option. The only file system affected will be the one for which the child **fsck** process was being created; no check will occur.

Internal Software Error: Cannot seek to block B — aborting

A disk seek to block number *B* has failed; this should never happen. Contact your Data General support representative if this message is displayed. The **fsck** program terminates.

Invalid response; please answer yes or no

An invalid answer has been entered in response to one of **fsck**'s questions. The **fsck** program will not continue until a valid response has been entered. The following strings are valid responses: **y, Y, yes, YES, n, N, no** and **NO**.

Errors During fsck Invocation

Before starting to check a file system, **fsck** must parse its command line and determine which files to check. The following messages result from command line errors or information in the file **/etc/fstab**.

The directory D is the mount point for F

The **fsck** program has been given a directory *D* to check and has determined that *D* is the mount point for the file system *F*. This message is purely advisory.

The flags –y, –n, –p, –q and –S are all mutually exclusive

More than one of the above flags has been specified on the **fsck** command line. At most one of them is allowed. The **fsck** program will abort.

Unknown option: –C

An unknown option flag, *C*, has been specified on the **fsck** command line. Valid flags are as follows: **–l**, **–y**, **–n**, **–p**, **–q**, **–t**, **–D**, **–f**, **–s**, **–S**, and **–x**. When you give it an invalid option, **fsck** will abort.

Errors During fsck Initialization

Before a file system check can be performed, **fsck** must set up certain tables and open certain files. The following messages can result from errors during this phase.

Block B is invalid Inode Table Block — rewrite as empty block?

The inode table block *B* does not contain the proper self–identification information. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES	Fix this error condition by rewriting this block as an empty file node table block. Any inodes that formerly occupied slots in this block will be cleared.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Cannot determine disk size of F

The **fsck** program has been given a file system *F* to check, but the size of *F* cannot be determined. This should never happen. Contact your Data General support representative if this message is displayed. The **fsck** program will abort checking this file system.

Cannot find a readable copy of the superblock

Neither of the two copies of the superblock can be read. The **fsck** program will abort checking this file system.

Cannot find a valid copy of the superblock

Neither of the two copies of the superblock contain the required self–identification information. The **fsck** program will abort checking this file system.

Cannot open F for reading

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for reading. Check the mode of *F*. The **fsck** program will abort checking this file system.

Cannot open F for writing

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for writing. Check the mode of *F* and make sure that no disks containing the file system are physically write–disabled. The **fsck** program will abort checking this file system.

Cannot read superblock copy N

One of the two superblock copies cannot be read. The **fsck** program will attempt to use the other copy and continue.

F is not a regular file, block–special file, character–special file or valid mount point

The **fsck** program has been given a file system *F* to check, but *F* is not of the correct type. *F* must be a file of type ordinary, block–special or character–special, or else it must be listed in the file **/etc/fstab** as a valid mount point directory. The **fsck** program will abort checking this file system.

File system is too large to check

Stand–alone **fsck** cannot allocate enough memory for its internal tables to begin checking the file system. The **fsck** program will abort checking this file system. Bring up your system and use the **fsck** command instead.

File system size stored in superblock is incorrect (N1 blocks should be N2) — fix?

The superblocks contain an incorrect file system size figure. If run with the **–p** or **–q** options, **fsck** will automatically correct this. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the file system size to N2, the actual size of the disk containing the file system.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Invalid default Data Element Size exponent: N — fix?

The default data element size for files (stored in the superblocks as a base–2 logarithm) is invalid. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the default data element size's exponent to 4.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Data Element Size exponent: N — fix?

The default data element size for directories (stored in the superblocks as a base–2 logarithm) is invalid. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default directory data element size's exponent to 4.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Index Element Size exponent: N — fix?

The default index element size for directories (stored in the superblocks as a base–2 logarithm) is invalid. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default directory index element size's exponent to 0.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Index Element Size exponent: N — fix?

The default index element size for files (stored in the superblocks as a base–2 logarithm) is invalid. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default index element size's exponent to 0.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid Disk Allocation Region size: N blocks

The DAR size stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid first allocation threshold file size: N — fix?

The superblocks contain an invalid first allocation threshold file size (the number of blocks a file can allocate in its initial DAR before all subsequent allocations are made from a different DAR). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the first allocation threshold file size to the default limit for DARs of the size specified in the superblock.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid number of inodes per Disk Allocation Region

The number of inodes per DAR stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid second allocation threshold file size: N — fix?

The superblocks contain an invalid second allocation threshold file size (the number of blocks a file can allocate in a noninitial DAR before all subsequent allocations are made from a different DAR). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the second allocation threshold file size to the default limit for DARs of the size specified in the superblock.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

No check necessary for F

The file system F is already marked mountable and **fsck** was invoked with the **–x** flag. The **fsck** program will not check this file system.

Superblock copies differ; using newer copy

Both copies of the superblock are readable and both contain the required self–identification information, but they differ. The **fsck** program will use the first copy (which is guaranteed to be more recent) and continue.

Superblock copy N is invalid

One of the two superblock copies does not contain the required self–identification information. The **fsck** program will attempt to use the other copy and continue.

Superblock has invalid contents in reserved area — fix?

A copy of the superblock has nonzero contents in a reserved area. If running with the **–p** flag, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES	The superblock's reserved area is initialized so that it contains all 0s.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 1 – Check Blocks and File Sizes

This phase of **fsck** operation is concerned with inodes. The following messages result from errors in inode types, inode format, file sizes and the data element pointers and index element pointers that make up a file's structure.

Incorrect block count in Inode I (N1 should be N2) — fix?

The inode *I*'s count of the blocks it uses is incorrect. If run with the **–p** option, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting inode *I*'s block count to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims an invalid block (B) — clear bad pointer?

The inode I claims block B, which does not exist. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the nonexistent block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims a system block (B) — clear bad pointer?

The inode I claims block B, which is a system block (a bitmap block, file node table block, DAR entry table block or superblock). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the system block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the system block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has an Index Block (B) with invalid format — clear bad pointer?

The inode I claims block B as an index block, but block B does not contain the proper self–identification information. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the index block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid contents in its reserved area — fix?

The inode I does not contain the proper self–identification information. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by initializing inode *I*'s reserved area.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid fragment size exponent (N) — clear?

The inode I has a disallowed exponent representing the size of the file's fragment. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is of unknown file type (O) — clear?

The inode I is of type *O*, which is an unrecognized octal number. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is partially truncated — fix?

The inode *I*'s size is shorter than the number of blocks allocated to it. If run with the **–p** option, **fsck** will automatically complete the truncation. Otherwise, **fsck** will ask to complete truncating inode I.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by completing the truncation of inode I down to the size stored in the inode.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 1b – Resolve Duplicate Claims

When **fsck** finds a block claimed by two or more files, it rescans the file system to find the original claimant of that block. This section lists the error messages that result from settling the claim to the disputed block.

Inode I claims another file's blocks — clear?

The inode I claims some blocks that belong to another file. **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. This will result in the same question being asked about the next claimant of the disputed block. As long as enough files are eventually cleared to resolve the duplicate claims on block B, **fsck** will continue normally. However, if at the end of Phase 1b any duplicate claims still exist, **fsck** will not mark this file system as mountable upon completing the check.

Errors During Phase 2 – Check Directory Contents

This phase is concerned with the contents of directories. The messages in this section result from improperly formatted directory blocks, an improperly formatted root directory, and bad directory entries. During this phase, all bad entries and inodes are removed from the file system tree.

Directory inode I has a hole — fix?

The directory inode *I* has at least one "hole" in its file structure (gaps before the end of file). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rearrange the directory blocks to fill in the hole.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rearranging the blocks in the directory to eliminate the hole.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has incorrect child count (N1 should be N2) — fix?

The directory inode *I*'s count of children, *N1*, is incorrect. If run with the **–p** or **–q** options, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the child count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting inode *I*'s child count to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has an invalid block (B) — rewrite as empty block?

The directory inode *I* has a block (address *B*) which does not contain the proper self–identification information. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES Fix this error condition by rewriting block *B* as an empty directory block. Any directory entries that formerly occupied this block will be destroyed.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 of invalid size — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry is too long, too short, or is not a multiple of 4 bytes in size. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 which is out of order — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2* which has a bad sequence number, meaning that the entry is invalid. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with filename of invalid size — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's file name is too long or too short. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with an illegal filename: F — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name *F* is **.** (dot) or **..** (dot–dot). These two names are reserved for the directory's links to itself and to its parent, respectively. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry? prompt are:`

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has a filename with an illegal character, octal value O — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name includes the illegal character *O*. A character is disallowed if it is non–ASCII or it is the slash character. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has an illegally long pathname — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the pathname for that entry relative to the root of the file system would exceed **MAXPATHLEN** (1024) bytes. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has invalid contents in its reserved area — fix?

The directory inode *I1* has a directory entry for inode *I2*, which has nonzero information in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the contents of the reserved area of inode *I2*.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by initializing the reserved area of inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is invalid — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not a valid inode number. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is unallocated — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not an allocated inode. If run with the **-p** option, **fsck** will automatically remove the directory entry for inode *I2*. Otherwise, **fsck** will ask to remove the directory entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to directory inode I2 — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a directory which does not list *I1* as its parent directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to symbolic link inode I2 — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a symbolic link which already has another hard link. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has an entry (for inode I2) which crosses a control point directory boundary — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I1* and *I2* have different space parent control point directories. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect filename length in directory inode I1 for directory inode I2 (N1 should be N2) — fix?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name length, *N1*, is incorrect. If run with the **–p** option, **fsck** will automatically correct the directory entry's name length to *N2*. Otherwise, **fsck** will ask to correct the name length.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the directory entry's length to *N2* bytes.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is of wrong file type — fix?

The root inode (inode 2) is not a control point directory. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the incorrect file type.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the file type of inode 2 to type control point directory.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is not allocated — fix?

The root inode (inode 2) is not allocated. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to allocate inode 2.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by allocating inode 2 as the root.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's parent directory is not the root — fix?

The root inode's parent directory is not the root (itself). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own parent.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's parent directory to itself.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space parent control point directory is not the root — fix?

The root inode's space parent control point directory is not the root (itself). If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own space parent.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's space parent control point directory to itself.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space usage limit is too large (N1 should be N2) — fix?

The root inode's space usage limit, *N1*, is bigger than the size of the file system, *N2*. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset the limit to *N2* blocks.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's space usage limit to *N2* blocks.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 3 – Check Connectivity

Phase 3 of **fsck** deals with the reconnection of unreferenced files and directories onto the file system tree. The messages in this section result from attempts to connect unreferenced files into the **lost+found** directory. Note also that any of the Phase 2 messages may be seen in this phase, as the contents of any reconnected directories must be checked.

Cannot find enough contiguous free blocks to expand directory inode I

The **fsck** program could not find enough contiguous free blocks to expand the directory inode I. Some unreferenced files may not reconnected as a result of this failure; they can be reconnected during a later **fsck** session after enough space has been freed in the file system.

Control point directory inode I has an entry named 'lost+found' which is not a directory — remove bad directory entry?

The control point directory inode I already has an entry named **/lost+found**, but which is not of type directory. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the bad entry from inode I. The bad entry's inode will itself be reattached in the new **/lost+found** directory which will be created in directory I1.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Could not reconnect inode I

The **fsck** program was unable to reconnect the unreferenced inode I because it could not allocate enough blocks to expand the **/lost+found** directory, or because it could not allocate a free inode to use as the **/lost+found** directory.

Directory inode I is already as large as it can become

The **fsck** program has discovered that a directory it was attempting to expand is already the maximum size a directory can become.

Inode I1 lists as its space parent inode number I2, which is not a valid control point directory — reset space parent to root?

The inode I1 has the noncontrol point directory inode I2 listed as its space parent. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset I1's space parent to inode 2, the root of the file system.

Possible responses to the `reset?` prompt are:

YES	Fix this error condition by resetting I1's space parent to inode 2, the root of the file system.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I needs to be expanded — fix?

The directory inode I needs to be expanded so that another directory entry can be added to it; I is either the root directory or the **/lost+found** directory. If run with the **-p** or **-q** options, **fsck** will automatically attempt to expand the directory. Otherwise, **fsck** will ask to expand it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by attempting to expand inode I.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I is unreferenced — clear?

The inode I has no links in the file system and an earlier reconnection failed or was refused.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I. The contents of the file will be destroyed.
NO	Ignore this error condition. Inode I will remain unattached and can be reattached during a later **fsck** session provided that enough blocks and/or inodes are free.

Inode I is unreferenced — reconnect?

The inode I has no links in the file system. If run with the **-p** option, **fsck** will automatically attempt to reconnect the file. Otherwise, **fsck** will ask to reconnect it.

Possible responses to the `reconnect?` prompt are:

YES	Fix this error condition by reconnecting inode *I* in the **/lost+found** directory, with the name "#*N*", where *N* is the inode number of I.
NO	Ignore this error condition.

The lost+found directory inode I already has an entry named 'F' — remove bad directory entry?

The **/lost+found** directory inode I has discovered that it already has an entry of the name F when it was trying to reconnect an unreferenced file which would have had the same name. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the spurious entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the entry for F; the inode referred to by that entry will be reattached with a name constructed from its inode number.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 4 – Check Link Counts and Resource Accounting

This phase checks the link counts of individual inodes and the resource counts (blocks and inodes used) of control point directories. The messages result from errors in these counts.

Control point directory inode I has incorrect inode allocation count (N1 should be N2) — fix?

The control point directory inode I has a bad count of the inodes used by it and all its space descendants. If run with the **–p** or **–q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the inode count for inode *I* to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Control point directory inode I has incorrect space allocation count (N1 should be N2) — fix?

The control point directory inode I has a bad count of the blocks used by it and all its space descendants. If run with the **–p** or **–q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the space count for inode I to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has incorrect link count (N1 should be N2) — fix?

The inode I has a bad link count. If run with the **–p** or **–q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the link count for inode I to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 5 – Check Disk Allocation Region Information

This phase deals with the disk allocation regions. Messages in this section result from errors in the components of the DARs: the bitmap, the free inode list, and various resource counts.

Block B of the Disk Allocation Region Information Area is invalid — fix?

The disk allocation region information area block B does not contain the proper self–identification information. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting this block as an empty disk allocation region information area block. The DAR information in the block will be corrected later in this Phase.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect Bitmap — fix?

The bitmap for DAR N is incorrect. If run with the **–p** option, **fsck** will automatically correct the bitmap. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the bitmap correctly.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect count of blocks used (N1 should be N2) — fix?

The block count for DAR *N* is incorrect. If run with the **–p** or **–q** options, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by changing DAR *N*'s block count from *N1* to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect counts of directories and inodes used — fix?

The counts of used files and directories for DAR N are incorrect. If run with the **–p** or **–q** options, **fsck** will automatically correct the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the counts of used inodes and directories correctly.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect free inode list — fix?

The linked list of free inodes in DAR number N is incorrect: it contains allocated inodes, duplicates, or it does not contain some inodes which are actually unallocated. If run with the **–p** or **–q** options, **fsck** will automatically correct the free list. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the free list for DAR number N.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has invalid contents in its reserved area — fix?

Disk allocation region number N has nonzero contents in its reserved area. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to zero out the reserved area.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by initializing the contents of the reserved area.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect summary counts in superblocks — fix?

The counts of used blocks and files in the two copies of the superblock are incorrect. If run with the **–p** or **–q** options, **fsck** will automatically correct the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the counts of used blocks and files correctly.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Advisory Messages During File System Cleanup

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system.

File System is now mountable

The **fsck** program has successfully completed checking the file system and it has been marked as mountable.

File System is still inconsistent and not mountable

The **fsck** program has completed checking the file system, but inconsistencies remain and the file system is still marked as unmountable. Re–run **fsck** in order to fix the remaining inconsistencies.

N1 of N2 blocks used (N3 free); N4 of N5 inodes used (N6 free)

The indicated number of blocks and inodes have been used, leaving the indicated number unallocated.

Unconnected files still remain. Mount the file system and remove files to free data blocks and inodes

The **fsck** program has successfully completed checking the file system and it has been marked as mountable. However, there are still unreferenced files in the file system. These unreferenced files can be recovered by running **fsck** again after enough blocks and inodes have been freed to allow them room to be reconnected.

Error Messages Exclusive to Fast Recovery Checking

This section lists errors that can only appear during checking of a fast recovery file system, one for which **fsck** logging was turned on. These messages do not appear with phased **fsck** messages in the previous section because fast recovery checking does not occur in phases.

While normal phased checking will return only the messages in the previous section, fast recovery file system checking may return the messages in this section as well as many (but not all) of those appearing in the previous section.

Bitmap block N in Disk Allocation Region N has an invalid format

This message is printed when **fsck** discovers a bitmap block with a bad self–id. The arguments are the lda of the bitmap block and the number of the DAR it belongs to.

Block N is invalid Inode Table block

This message is printed when an Inode Table block fails to self–ID. The numeric argument is the invalid block.

Block N of the Disk Allocation Region Information Area is invalid

This message is printed when **fsck** discovers a DARIA which does not self–ID. The argument is the ordinal number (within the DARIA) of the faulty block.

Cannot allocate memory for internal tables (N bytes requested)

This message is printed when an attempt to allocate memory fails. The numeric argument is the number of bytes requested.

Cannot find enough contiguous free blocks to allocate a missing index element for inode I

This message is printed when **fsck** fails to allocate an index element. The argument describes the inode in question.

Cannot open fast recovery dump file F

This message is printed when the open of the file to hold the human–readable version of the fast recovery log fails. The argument is the pathname of the dump file.

Cannot read DAR table

This message is printed when the DAR table cannot be read.

Cannot read fast recovery log at lda B

This message is printed when the read of one of the fast recovery disk logs fails. The argument is the logical disk address of the log.

Cannot recover from log. Run full fsck

This message is printed when a fatal error happens while trying to recover with the log.

Cannot write DAR table

This message is printed when the DAR table cannot be written.

Directory entry D in Directory inode I contains the wrong inode number (B should be B)

This message is printed when a directory entry is discovered to contain an incorrect file node number. The arguments are the directory entry file name, the file node number of the directory containing the entry, the entry's incorrect file node number, and the correct file node number.

Directory inode I has a spurious entry for file name F with file node number B

This message is printed when an unneeded directory entry is found. The arguments are the directory's file node number, the name of the missing entry and the file node number that should be in that entry.

Directory inode I has an incorrect parent inode number (B should be B)

This message is printed when a directory file node containing the wrong parent inode number in its din is found. The arguments are the directory's inode number and the incorrect and correct parent inode numbers.

Directory inode I has incorrect child count (N should be N)

This message is printed when **fsck** discovers an erroneous child count in a directory. The string argument describes the inode in question. The first numeric argument is the old, incorrect count; the second numeric argument is the correct count.

Directory inode I is missing an entry for file name F with file node number B

This message is printed when a directory entry is found to be missing. The arguments are the directory's file node number, the name of the missing entry and the file node number that should be in that entry.

Disk Allocation Region N has an incorrect Free Inode List

This message is printed when **fsck** discovers a DAR with incorrect Free File Node List. The first argument is the DAR number.

Disk Allocation Region N has incorrect Bitmap

This message is printed when **fsck** discovers an incorrect bitmap. The argument is the number of the DAR with the bad bitmap.

Disk Allocation Region N has incorrect count of blocks used (N should be N)

This message is printed when **fsck** discovers a DAR with incorrect space count. The first argument is the DAR number. The second argument is the incorrect, old count. The third argument is the correct count.

Disk Allocation Region N has incorrect counts of directories and inodes used

This message is printed when **fsck** discovers a DAR with incorrect directory or file node counts. The first argument is the DAR number.

File node and space accounting partially recovered. Run full fsck to recover accounting information

This message is printed when **fsck** completes but accounting recovery from the log was not complete. This error does not prevent the file system from being mounted, but at some point, full **fsck** should be run to fully correct accounting information.

Inode I has a bad fragment size for data element #0 (B should be B)

This message is printed when a file node's `fragment_size_exponent` field is found to contain an incorrect value. The arguments are the inode number, the bad fragment size exponent and the correct fragment size exponent.

Inode I has an Index Block (N) with invalid format

This message is printed when **fsck** discovers an element pointing to an index block which does not self–ID. The string argument describes the inode in question. The numeric argument is the bad block's number.

Inode I has an incorrect file size (N bytes should be N bytes)

This message is printed when **fsck** discovers a node which has an incorrect file size. The string argument describes the inode in question. The first numeric argument is the old, incorrect size. The second numeric argument is the correct size.

Inode I has an incorrect space parent inode number (B should be B)

This message is printed when a file node containing the wrong space parent inode number is found. The arguments are the file's inode number and the incorrect and correct space parent inode numbers.

Inode I has bad element pointers

This message is printed when an element pointer in the file node is found to contain the incorrect address and the element is allocated. The argument is the inode number.

Inode I has incorrect inode allocation count (N should be N)

This message is printed when **fsck** discovers a node which has an incorrect node count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I has incorrect link count (N should be N)

This message is printed when **fsck** discovers a node which has an incorrect link count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I has incorrect space allocation count (N should be N)

This message is printed when **fsck** discovers a node which has an incorrect space count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I is incorrectly marked as allocated

This message is printed when a file node allocation has to be backed out or the results of a deallocation did not make it to disk. The argument is the file node number.

Inode I is partially truncated

This message is printed when **fsck** discovers a partially truncated file. The string argument is the file in question.

Internal software bug O

This message is printed when a sanity check fails, indicating a software bug. The argument is a status encoded with FSCK_ENCODE_BUG_STATUS.

Missing log entry for file system fnode accounting

This message is printed when it is discovered that there are file node allocations or frees on the file system, but that no log entry giving the initial value of the file system's `number_of_used_file_nodes` was made. This error indicates that there is a bug in the kernel logging code.

System buffers containing data from the following files may not have been written to disk:

This message is printed when **fsck** fails to allocate an index element. The argument describes the inode in question.

Unexpected child count N in inode I

This message indicates a bug in the log recovery code.

Unknown kind of log entry

This message is printed when an unrecognized type of log entry is found in the log.

End of Appendix

Appendix C
SAF Reference Cards

This appendix contains tables that you may copy or remove and use as reference cards for the Service Access Facility (SAF). For information on SAF, see Chapter 9.

Port Monitor Management Reference Card

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.

Licensed Material – Property of copyright holder(s)

Service Administration Reference Card

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 Reference Card

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**.

Licensed Material – Property of copyright holder(s)

listen Reference Card

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 Appendix

Appendix D
Directories and Files

This appendix lists the directories and files of the DG/UX system that are of interest to a system administrator. For a list of all directories and files shipped with the DG/UX system, look in the directory **/usr/release/dgux_*.fl**. For detailed information on any of the entries here, see the relevant manual page.

First, this appendix discusses the **root**, **var**, **usr**, and **srv** directories. Then it discusses some files of interest to a system administrator. Notice that some files and directories in the DG/UX system have *physical* locations and *logical* locations. A physical location is a file's real location. A logical location is a symbolic link. For instance, when you reference **/usr/spool/lp** (logical), you are actually referencing **/var/spool/lp** (physical). In this appendix, we denote a symbolic link by enclosing the name in parentheses.

Contents of the Root Directory

The **root** logical disk is mounted on the directory **/**. We refer to those files on the **root** logical disk as "being in root." There are physical and logical directories on the **root** logical disk. They are:

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

/admin This 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 directory also contains the **crontabs** directory.

The **/admin/crontabs** directory contains prototype **crontab** files useful to the system administrator. The **root.proto** file contains miscellaneous **cron** jobs for maintaining the system. These jobs perform a variety of functions such as cleaning out temporary file directories and truncating logs. See Chapter 2 for more information. The **lp.proto** file contains jobs for maintaining the LP subsystem. The **uucp.proto** files perform functions for the UUCP file transfer/remote command execution utility. Chapter 2 discusses the **cron** facility and the prototype **crontab** files in more detail.

(/bin) Symbolic link to **usr/bin**. Contains public commands.

/dev Contains device nodes, also called special files.

/dgux Executable kernel file.

/dgux.aviion
Default name of kernel executable file (typically a hard link to **dgux**, above).

/dgux.installer
Executable kernel used for installation of system software.

/dgux.starter
Hard link to **dgux.installer**, above.

/etc	Contains configuration files and system data.
(/lib)	Symbolic link to **usr/lib**. Contains object libraries.
/local	Contains site-specific files.
/opt	Parent directory for user–configurable portions of applications packages.
/proc	Currently unused.
/sbin	Contains minimum system commands to get the system up.
/srv	A mount point for the **srv** logical disk. Contains client and release management files. See "Contents of the /srv Directory."

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

/tmp	Used for temporary system files.
/usr	Mount point directory for the **/usr** file system. See the 7f>Contents of the /usr Directory."
/var	Used for system data files whose size varies as the system runs. See "Contents of the /var Directory."

Contents of the /var Directory

The **/var** directory contains files that are release dependent, have read and write permissions set, and are dynamically sized.

/var/adm	This directory contains a variety of files produced by the system, such as system error logger (**syslogd**(1M)) messages This directory contains the data collection files for the accounting system. See Chapter 14 for complete information on accounting system files and directories.
/var/Build	Kernel builds by **sysadm** Auto Configure and Build operations take place here.
/var/cron	Contains the **cron** log.
/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.

/var/spool	Contains spooling files for LP, UUCP, and **cron**.
/var/spool/lp	Contains all of the files and directories of the LP system. See Chapter 11 for complete information on these files and directories.
/var/spool/cron	Contains **cron** databases and dynamically–sized files.
/var/spool/uucp	Contains files specific to UUCP.
/var/preserve	Text editor file save area for sudden program halt.
/var/tmp	User temporary file space.
/var/ups	Contains logs and the status file for the Uninterruptible Power Supply **upsd**(1M) daemon.

Contents of the /usr Directory

The **usr** logical disk is mounted on the **/usr** directory. Files in this directory are release dependent and read–only. There are physical and logical directories on the **usr** logical disk. They are:

(/usr/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**(1) command.
/usr/etc	Contains database and configuration files.
/usr/etc/master.d	Contains master files. These files list devices and kernel parameters for the DG/UX system. This directory may also contain master files for other packages that have kernel components, such as TCP/IP and 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 14.
/usr/lib/gcc	Contains the DG/UX GNU C compiler. For information 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 an administrator.
/usr/sbin/init.d	The **init.d** directory contains executable files 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.
(/usr/share)	Symbolic link to **/srv/share**.
(/usr/spool)	Symbolic link to **/var/spool**.
/usr/src	Parent directory for source code.
/usr/src/uts/aviion/lb	Contains the kernel libraries which are used to build the kernel image. See Chapter 4. Also see **config**(1M) and **make**(1).
/usr/stand	Contains stand–alone utilities and bootstrap programs.
(/usr/tmp)	Symbolic link to **/var/tmp**.
(/usr/ucb)	Symbolic link to **/var/ucb**.

Contents of the /srv Directory

The **/srv** directory contains the directories and files needed for managing operating system releases and clients.

/srv/admin	Contains **sysadm** databases and information files.
/srv/admin/clients	Contains **sysadm** client data.
/srv/admin/defaults	Contains **sysadm** defaults for releases and clients.
/srv/admin/releases	Contains **sysadm** OS release data.
/srv/dump	Dump space on a one–per–client basis.
/tftpboot	Contains links to bootstraps for diskless clients.
/srv/release	Contains space for each release's **usr** and client roots.
/srv/release/PRIMARY	Contains symbolic links to the server's **usr** and **/** files.
/srv/share	Contains release independent shared software.
/srv/swap	Swap space on a one–per–client basis.

DG/UX Administrative Files

This section is not intended to be an exhaustive look at the DG/UX files; rather, it highlights the files that you'll be using more often than others. Subsections here are:

- Contents of the **/etc** Directory
- Administrative Commands in the **/sbin** Directory
- Administrative Files in the **/usr** Directory
- Administrative Files in 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 may alter most if not all of these files yourself using a text editor; nevertheless, we recommend that in cases where **sysadm** or another administrative utility offers an interface for managing the file, you choose the interface rather than the 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 will continue 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 **/etc/dgux.prototab**.

/etc Database Files Maintained by the System

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

/etc/devlinktab

This files contains entries used to make short–named links to device nodes with otherwise unwieldy names. The system creates this file at boot to reflect the current configuration of the system. 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**(1) command to access this information.

/etc/wtmp

The **/etc/wtmp** filecontains 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**(1) and **who**(1) commands to access this file. Periodically, this file should be cleared or truncated. See Chapter 2 for information on truncating this 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**(1M). It describes the media characteristics for each medium made available to **dump2**. See **dumptab**(4) 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. Chapter 7 discusses failover disks.

/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 ONC/NFS format; we recommend this format even if you are not using ONC/NFS.

```
/dev/dsk/root  /  dg/ux  rw  d  1
```

The above entry indicates a local file system mount, and the following entry indicates an ONC/NFS remote file system mount.

```
titan:/usr/titan  nfs  rw,hard  x  0
```

The **fstab** format was changed to support ONC/NFS file systems as well as local file systems. The old style **fstab** entries are also supported. See **fstab**(4) and Chapter 8 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, which are separated by a colon (:):

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

See "Adding Groups" in Chapter 13 and **group**(4) for more information. If you have ONC/NFS, see **yppasswdd**(1M) and your ONC/NFS documentation.

/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 Chapter 11.

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

See "The User's Environment" in Chapter 13 and **passwd**(4).

/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**. See Chapter 13.

/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**(1M), **at**(1), and **batch**(1). The **queuedefs** file, described in more detail in **cron**(1M), contains parameters governing the **at** and **batch** queues. See Chapter 2 and the **crontab**(1) 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) modifying this file during package setup. Chapter 3 summarizes the various run levels and describes their uses. See **inittab**(4) 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. See "The User's Environment" in Chapter 13 for more details.

/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 13 for examples.

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**(1M) during boot.

/sbin/fsck

Runs the **fsck**(1M) 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**(1) command.

/sbin/shutdown

Brings the operating system down to single–user mode (run level S). See Chapter 3.

/sbin/su

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

/sbin/umount

Unmounts a remote file system.

Administrative Files in the /usr Directory

The following files are available in the **/usr** directory.

/usr/sbin/init.d

This directory contains the system's check scripts and **rc** scripts.

/usr/sbin/setup.d/usr

This directory contains set up scripts that modify a host's **usr** space. Setup scripts might include those for TCP/IP and ONC/NFS.

/usr/src/uts/aviion/cf/system.*.proto

This file contains your custom version of the devices and configuration parameters listed in **/usr/etc/master.d/dgux**. For configuration, the **config**(1M) program runs on the **system** file and produces program code in **conf.c**. Prototype system files are shipped with software packages with kernel content. Prototype files have names of the form **system.*.proto.**

Administrative Files in the /var Directory

The following files are located in the **/var** directory. These files are useful for monitoring superuser activity and performing recurring system administrative tasks.

/var/adm/sulog

The **/var/adm/sulog** file contains a history of superuser (**su**(1)) command usage. As a security measure, this file should not be readable by others. The **/var/adm/sulog** file should be periodically truncated to keep the size of the file within a reasonable limit. For more information on truncating **sulog**, see Chapter 2.

A typical **/var/adm/sulog** file follows:

```
SU 08/18 16:16 + console smitht-root
SU 08/18 23:45 + tty00 jones-root
SU 08/19 11:53 + console smitht-sysadm
SU 08/19 15:25 + console root-sysadm
SU 08/19 23:45 + tty00 root-uucp
```

/var/cron/log

A history of all actions taken by **/usr/sbin/cron** is recorded in the **/var/cron/log** file. The **/var/cron/log** file should be periodically truncated to keep the size of the file within a reasonable limit. For information on truncating the **cron** log, see Chapter 2.

End of Appendix

Index

Licensed Material – Property of copyright holder(s)

Licensed Material – Property of copyright holder(s)

Licensed Material – Property of copyright holder(s)

TIPS ORDERING PROCEDURES

TO ORDER

4. 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

5. 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

6. 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

7. 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

8. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

9. Allow at least two weeks for delivery.

RETURNS

10. Items ordered through the TIPS catalog may not be returned for credit.
11. 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

12. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME	COMPANY NAME
ATTN:	ATTN:
ADDRESS	ADDRESS (NO PO BOXES)
CITY	CITY
STATE ZIP	STATE ZIP

Priority Code ____________ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side)	Title	Date	Phone (Area Code)	Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0–$149.99	0%
$150–$499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) ____________

ORDER TOTAL	
Less Discount See B	–
SUB TOTAL	
Your local* sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is________. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date ☐☐☐☐

Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508–870–1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Managing the DG/UX™ System

093–701088–02

Cut here and insert in binder spine pocket