Data General

Customer Documentation

Installing and Managing the DG/UX™ System

AViiON™

PRODUCT LINE

Addendum
Installing and Managing the DG/UX™ System

086-000161-00

The accompanying document replaces Chapter 2 of *Installing and Managing the DG/UX™ System* (093-701052). Technical changes in the chapter are marked by a revision bar in the right margin. Please remove your existing copy of Chapter 2 and replace it with this one.

This change also affects the Title and Notice pages. This package includes updated versions of these pages; please use them to replace the existing Title and Notice pages in your manual.

093-701052-01

Installing and Managing the DG/UX™ System

093-701052-01

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

Ordering No. 093-701052

Revision 01, December 1989

Installing and Managing the DG/UX™ System

093-701052-01

Revision History:	Effective with:
Original Release — April1989	DG/UX Rev. 4.10
Revision 01 — August 1989	DG/UX Rev. 4.10
Addendum 086-000161 — December 1989	DG/UX Rev. 4.20

Preface

This manual is for system administrators who are familiar with the UNIX® operating system. This manual shows you how to do a first-time installation of the DG/UX™ operating system and how to operate and manage the system on a routine basis.

Are You Experienced?

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

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

Manual Outline

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

This manual is composed of the following chapters and appendixes:

Chapter 1	Part 1: Managing a server or client OS with **sysadm**. How to use sysadm menus for managing the OS of any AViiON machine regardless of whether it is a stand-alone, server, or client machine. Part 2: Managing the server/client group (servnet) environment. Focuses on what tasks need to be done and how to do them; not dwelling on "who" does what task. Individual host administrators in the servnet negotiate who will do what tasks. Explains concepts and relationships: servers, clients. Shows the sequence of a diskless client booting an operating system from an OS server.
Chapter 2	Planning and loading the DG/UX system. Shows how the DG/UX file system has been reorganized to support the server/client environment. Setting up a stand-alone or server host. Setting up diskless clients on a server.

Chapter 3	Operating the DG/UX system. Startup and shutdown, recovering from system trouble (e.g., crashes, power outages). Explanation of rc scripts and run levels.
Chapter 4	Reconfiguring the system on a routine basis. Setting tuneable configuration parameters. Setting system date and time.
Chapter 5	Managing releases, loading software, listing information about releases.
Chapter 6	Managing client machines. Setting defaults, adding clients, deleting clients.
Chapter 7	Using the disk management utility diskman to format physical disks, create logical disks, create file systems, check file systems, display physical disk information, update the operating system, and do various other disk tasks.
Chapter 8	File system management. Shows how to mount, unmount, add, delete, and make backup tapes.
Chapter 9	File information. Finding and displaying information on files and file systems according to age, size, name, and block use.
Chapter 10	Setting up and managing terminals.
Chapter 11	Managing line printers and laser printers. Start/stop scheduler, set defaults, cancel, enable/disable, add/delete printers, etc.
Chapter 12	Setting up UUCP files and directories.
Chapter 13	Network terms and definitions. Basic network management functions: setting parameters for NFS® and TCP/IP, adding/deleting hosts and networks.
Chapter 14	Adding user accounts, creating aliases and groups. Explains how these are used and managed differently in stand-alone and servnet situations.
Chapter 15	Monitoring how system resources are being used with accounting programs. Printing summary reports on system use.
Appendix A	Device names and codes for servers and workstations.
Appendix B	Directories and files. Descriptions of the directories and files that are shipped with the DG/UX system.
Appendix C	Error messages: LP, UUCP, and errno error messages.
Appendix D	The **fsck** file system check and repair program: invoking, options, arguments, output, and error conditions.

Appendix E	Expert UUCP information. Supplements information in Chapter 12.
Appendix F	Glossary

Related Manuals

- *System Manager's Reference for the DG/UX™ System* (093-701050). This manual contains all the detailed descriptions (manual pages) for commands relating to system administration or maintenance.
- *Managing NFS® and Its Facilities on the DG/UX™ System* (093-701040). This manual is for users, system administrators, and programmers. It describes the user interface, system administration, and internal protocols used by the Network File System (NFS).
- *DG TCP/IP (DG/UX™) User's Manual* (093-701023). This manual introduces Data General's TCP/IP (DG/UX) family of protocols and describes how to use the package.
- *Using the DG/UX™ System* (093-701035). This manual is for all users. It describes in detail the primary tasks that a user is likely to perform in a DG/UX environment. This manual includes short tutorials on commonly used commands and shell scripts.
- *Using the DG/UX™ Editors* (093-701036). This manual is for all DG/UX system users. It describes how to use the DG/UX text editors including **vi**, **sed**, **editread**, and **ed**.
- *Using the AViiON™ System Control Monitor (SCM)* (014-001802). Describes how to use the SCM commands and menus to debug programs, control program flow, and boot media on AViiON RISC-based hardware.
- *Setting Up and Starting AViiON™ 5000 Series Systems* (014-001806). Describes how to unpack, check, and install system components and connect options. Explains how to power up, run diagnostics, and load the operating system. Includes operational, physical, electrical, and environmental specifications of the computer unit.
- *Setting Up and Starting AViiON™ 300 Series Stations* (014-001801). Describes how to unpack, check, and install system components and options. Explains how to power up, run diagnostics, and load the operating system. Includes operational, physical, electrical, and environmental specifications of the computer unit, monitor, keyboard, and mouse.

Readers, Please Note

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

Convention	Meaning
boldface	In command lines and format lines: Indicates text (including punctuation) that you type verbatim from your keyboard. All DG/UX commands, 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.
italic	In format lines: Represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands.
[*optional*]	In format lines: These brackets surround an optional argument. Don't type the brackets; they only set off what is optional. The brackets are in regular type and should not be confused with the boldface brackets shown below.
[ ]	In format lines: Indicates literal brackets that you should type. These brackets are in boldface type and should not be confused with the regular type brackets shown above.
...	In format lines and syntax lines: Means you can repeat the preceding argument as many times as desired.
`$` and `%`	In command lines and other examples: Represent the system command prompt symbols used for the Bourne and C shells, respectively.
↵	In command lines and other examples: Represents the New Line key. Note that on some keyboards this key might be called Enter or Return instead of New Line.
< >	In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as <**Ctrl-D**>, <**Esc**>, and <**3dw**>) from surrounding text.

Contacting Data General

If you have comments on this manual, please use the prepaid Comment Form that appears after the Index.

Service assistance on Data General software or hardware is available via telephone in the United States and Canada. Ask your Data General representative for the number. Outside North America contact your local Data General office.

End of Preface

Contents

Appendix D — File System Checking: fsck

Appendix E — Expert UUCP Information

Appendix F — Glossary

Tables

Figures

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

Welcome. You have opened this manual because you want to manage a DG/UX™ system. In revisions of the DG/UX system before 4.10, we described system administration tasks as being done by one person on a single machine that supported users on a traditional multiuser system.

DG/UX system administration has changed.

Now, the DG/UX operating system also supports diskless client machines in an OS server-client environment. An OS server machine uses the Transmission Control Protocol/Internet Protocol (TCP/IP) and the Network File System (NFS®) to provide an operating system to remote client machines. This scenario involves several sets of tasks, which can be done by the OS server administrator or shared among OS client administrators.

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

- **Stand-alone machine** -- this could be a multiuser system that supports conventional CRTs or a workstation that boots from its own disk.
- **OS Client machine** -- a workstation that boots its operating system from an OS server machine via TCP/IP and NFS.
- **OS Server machine** -- a machine that provides a bootable DG/UX image and file system space to diskless client machines via TCP/IP and NFS.

How Chapter 1 is Organized

This chapter is divided into parts One and Two.

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

Part Two introduces the concept of a servnet and discusses the aspects of the server and client relationship.

Administrative Roles

We've mentioned the kinds of machines you might be managing, but we haven't addressed what the tasks might be for each manager. In order to present a recommendation for doing these tasks, we'll refer to three machines: a stand-alone, an OS server, and an OS client. The next three sections address what tasks will be done on which machine. The following is meant as an example; tasks can be negotiated and distributed as managers choose.

Tasks on Stand-Alone and OS Server Machines

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

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

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

Tasks on a Data General OS Client

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

Tasks that can be done from the OS client are

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

Tasks on a Foreign OS Client

A Data General OS server can provide operating system service to a foreign OS client, but the client is largely responsible for its own administration. Notable among the foreign client's responsibilities are reconfiguring the kernel, setting up and managing networking software, and using its own operating system documentation. To take advantage of the **sysadm** utility for foreign clients, you must set up a **srv** directory structure for the foreign client systems. Chapter 2 shows how you can do this in "File System Mapping: DG/UX to Foreign."

Part One: Managing with sysadm

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

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

Help?

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

sysadm and diskman

In addition to **sysadm**, you will be using another utility called **diskman**, which comes in two versions. Site without preloaded disks will use the *stand-alone* version during installation to load the starter system components. Stand-alone **diskman** is in the **/usr/stand** directory. Later, you'll use the *stand-among* version for tasks such as creating logical disks, creating file systems, deleting logical disks, and obtaining information on physical and logical disks. You invoke this utility via the **sysadm diskmgmt** command. In **diskman**, you choose from menus structured similar to those of **sysadm**. See Chapter 7 for detailed information on **diskman**.

Using sysadm Commands

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

```
                      SYSADM MAIN MENU

 1 diskmgmt           Enter the diskman program
 2 sysmgmt            System configuration management menu
 3 fsmgmt             File system management menu
 4 fileinfo           File information menu
 5 ttymgmt            TTY management menu
 6 lpmgmt             Line Printer management menu
 7 usermgmt           User management menu
 8 uucpmgmt           UUCP management menu
 9 networkmgmt        Network management menu
10 releasemgmt        Release Management menu
11 clientmgmt         Client Management menu

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

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

Table 1-1 How to Use sysadm Menus

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

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

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

The sysadm Menu Set

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

1 diskmgmt

```
                Diskman Main Menu

1 Physical disk management menu
2 Logical disk management menu
3 File system management menu

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

2 sysmgmt

```
                    System Configuration Management

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

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

3 fsmgmt

```
                        File System Management

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

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

4 fileinfo

```
                         File Information

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

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

5 ttymgmt

```
                    TTY Management

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

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

6 lpmgmt

```
                        Line Printer Management

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

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

7 usermgmt

```
                    User Management

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

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

8 uucpmgmt

```
                    UUCP Management

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

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

9 networkmgmt

```
                    Network Management

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

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

10 releasemgmt

```
                 Software Release Management

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


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

11 clientmgmt

```
                   Diskless Client Management


 1 addclient        Add a diskless client entry
 2 delclient        Delete a diskless client entry
 3 lsclient         List information about diskless clients
 4 clientdefaults   Create or modify a set of diskless client defaults
 5 bootdefault      Change the default release for a client

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

Invoking sysadm with Arguments

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

# sysadm sysmgmt ↵

Your screen displays the following:

```
                   System Configuration Management

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

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

At this level, **sysadm** waits for your next choice. But perhaps you want to go even farther. You need not stop here if you already know you want to go to the **datetime** function. In this case, you can simply type:

# sysadm datetime ↵

When you use a direct command like this, you avoid walking through two menus, and the system tells you *where* you are within the **sysadm** scheme:

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


The current time zone is EST. Daylight Savings Time is used.

The current date and time are: 05/10/89 08:35.
```

Next, you will be asked what time zone you'd like. The default is the current time zone, in this case Eastern Standard Time, as indicated by number 3 in brackets below. If you'd like to change to another time zone, type **?** to see a list of time zones. For now, let's say that you want the default time zone. As you respond to each query line, another is printed. Below, we show how repeated New Lines advance us through each **datetime** query, accepting the current defaults.

```
Time Zone? [3] ↵
Does your area use Daylight Savings Time? [y] ↵
Month? [05] ↵
Day of the month?[10] ↵
Year? [89] ↵
Hour? [08] ↵
Minute? [36] ↵

No changes were entered.

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

The sysadm Pager

When **sysadm** prints more than a screen full of text, you'll need to press the space bar to display the next page of text. You'll see the following when a function uses the pager:

```
Next Page?
```

Remember, anytime you are unsure of how to respond to a query, type **?** to display HELP message.

Shell Escapes

You can escape to the shell while in the middle of using a **sysadm** function. When you conclude your command, you automatically return to where you were in **sysadm**. And you continue as before. If you use the **sh** command with **!**, you'll spawn a new shell and **sysadm** continues to run in your previous shell. So if you use **!sh** to spawn a new shell, remember to use either the **exit** command or a Ctrl-D to return to the **sysadm** you left running in the previous shell. *Do not* type the **sysadm** command from the new shell you've spawned; this will create another image or copy so that *two* **sysadm** programs will be running.

Using Procedures

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

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

Command References

Sometimes we will refer you to manual reference pages, or man pages. Each one describes a DG/UX command, system call, file, or program. A man page has a *name* and a *section number*. Here's what a man page reference looks like:

wall(1M)

A **wall** is a command that means "write to **all**", a way of sending a broadcast message to all users. The (1M) tells you in what reference manual the command is found. The numbered sections are located in the following manuals:

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

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

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

```
$ man wall ↵
```

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

DG/UX Expert Sections

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

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

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

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

Part Two: Managing Servers and Clients

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

Terms

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

host	refers to any machine: stand-alone, server, or client.
stand-alone	a machine with its own disks. Some stand-alones support dumb terminals in a traditional multiuser environment where all terminals are running the same release. A stand-alone does not need TCP/IP or NFS to service its terminals, and it has no OS clients. A workstation with its own disk is also a stand-alone.
OS server	refers to a host providing an operating system and disk space for client machines. Servers can be homogeneous or heterogeneous.
	A homogeneous OS server provides a *single system release* to many other clients which all share a single copy of some of the system release files (**usr**), but which also maintain private copies of other system release files (**root**). Processing is done on each client's processor.
	A heterogeneous OS server provides *many system releases* to many clients. Clients share release software (**usr**) and maintain private files (**root**) the same way clients of homogeneous OS servers do; clients have the ability to boot other OS releases supported by the server if the other releases are compatible with client hardware.
	Note that there are other kinds of servers besides OS servers. A Yellow Pages (YP) server provides YP database information to YP clients. An NFS server provides file system access to remote NFS clients. When the term *server* is used unmodified it refers to an OS server.
client	a host which gets its system files from a disk connected to an OS server. When the term *client* is used unmodified it refers to an OS client.

diskless client	an OS client with no disks of its own. An OS client may have its own local disk, but uses an OS server for its system software (the traditional **/** and **usr** directories).
workstation	a system with its own processor, its own graphics terminal, and graphics software (shared or host dependent). A workstation could be an OS server, or an OS client with or without disks.
servnet	the collective unit formed by an OS server, its clients, and its releases. For example, a server supplying OS releases for two Data General workstations and one foreign workstation would be a servnet.
directory tree	refers to the **/**(root), **/usr**, **/srv**, and other directories that make up the directory space of a DG/UX system. Since diskless clients in DG/UX 4.10 allow **/**, **/usr**, and **/srv** to be subdirectories in the server's file system, the terms **root** file system and **usr** file system are no longer appropriate when referring to a client. Clients use directories on the server for these functions instead of having their own file systems. The OS server has a **root** logical disk file system and a **usr** logical disk file system which contains the software associated with the primary release. (The primary release resides in **/srv/release/PRIMARY**.) By logical disk file system, we mean a file system directly associated with a specific logical disk; the general case of one logical disk = one file system. See Chapter 7 for more on logical disks.
mapping	associating the elements of two different representations of a system (like a directory tree) so that a correspondence exists between the two systems. Every element in one system can be mapped to an element in the other system. The **sysadm** program maps all of a host's **root** and **usr** directory trees through the **/srv** directory tree. Mapping all **root** and **usr** directory trees in a single, consistent way eliminates the need for conversions if a host's role changes from stand-alone, server, or client. Because information describing the OS clients and releases is kept in the **/srv** directory tree, OS server hosts should create a separate logical disk named **srv**. Within the **srv** directory tree, the directory **PRIMARY** is used to manage primary releases. The directory **MY_HOST** is used to manage a host's **root** directory tree. Both of these names are reserved and should not be used elsewhere.
release dependent	the system commands and files that are dependent on a release. A file that can't or shouldn't be shared between releases is *release dependent*. If, for example, man pages and certain ASCII data files can be shared by more than one release, then they are *release independent*. The master files and most commands are examples

	of release dependent files. Release dependent files may or may not be host dependent.
host dependent	the commands and files that are dependent or unique to an individual host. If a given file can't or shouldn't be shared between several hosts, then it is *host dependent*. Every client host needs its own copy of host dependent files and needs to be able to write to its own data files. This class of files also contains the set of commands and data files required to boot a host. The **/etc/passwd** file and **.profile** are examples of host dependent files. Host dependent files may or may not be release dependent.
package	a set of software traditionally called a product.
release	a set of software intended for a specific architecture and operating system. A release encompasses all software required for a host. A release identity is defined by the contents of the release's **/usr** directory. Additional software packages loaded into the release's **/usr** directory become part of the release.
composed release	multiple products packaged on a single tape set that can be installed by **sysadm**. Each product tape has its own install and setup routines that can be executed at load time.
primary release	the release that resides in the server's **root** and **usr** logical disk-file systems. Other (secondary) releases may reside on the server in a separate directory. The primary release resides in **/srv/release/PRIMARY**.
foreign release	a release supplied by a vendor other than Data General. Releases supplied by Data General are called native releases.
tapeless	an OS server without a tape drive. To read a release tape, a tapeless server must access another host that does have a tape drive.

Managing in the Servnet

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

The OS Server

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

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

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

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

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

The OS Client

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

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

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

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

Windows

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

How an OS Client Boots

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

1) The OS server manager must get the Ethernet address and host name of the OS client wishing to boot.

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

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

4) The OS client host powers up and sends out its Ethernet address in a broadcast message. The OS server host recognizes the client's Ethernet address and sends back the client's Internet address. The client host sends another message which generates a boot file name in the server hosts's **/tftpboot** directory. The OS server host sends a boot program to the client.

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

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

7) The client uses NFS and mounts the remote **root** file system, and opens and reads the kernel binary file. The kernel forks a shell to read the system startup file and the client comes up in a default run level.

TCP/IP, NFS, and YP

The servnet relies on TCP/IP and NFS to provide operating system service to diskless clients. These products are shipped as part of the OS server's release package. This manual will show you the parameters we'll use for TCP/IP and NFS that apply to a sample installation. This manual does not provide any detailed information on these networking products. If you need conceptual information on these products, see *Installing and Managing DG TCP/IP (DG/UX™)* and *Managing NFS® and Its Facilities on the DG/UX™ System* before you attempt to install either product.

Although the Yellow Pages (YP) are not absolutely necessary for managing a servnet, it is a facility that provides an efficient database method for tracking and servicing users and machines. We'll use YP in our example system and we recommend that you use it also. *Managing NFS® and Its Facilities on the DG/UX™ System* gives information on setting up YP on a server or client host. Note that the Yellow Pages can be managed from any host on the servnet. For convenience, we'll manage the YP from our OS server, the master server in our example system.

End of Chapter

Chapter 2
Installing the DG/UX System

Let's get started. Installation information in this chapter applies to stand-alone multiuser systems, stand-alone workstations, or OS servers supporting diskless workstations. Some material in this chapter is useful if you're installing an update of the DG/UX system on an already-installed system; however, the chapter is primarily intended to help you through the initial installation. If you are installing an update on an existing DG/UX system, refer to the release notice provided with the software.

As we go through setting up our example system, you'll find that installing a stand-alone machine is only slightly different from installing an OS server to support diskless workstations.

In this chapter, we'll look at an installation checklist and an example hardware configuration that we'll use to install the operating system. Then, we'll take a brief look at the device specifications that you'll see recurring throughout this manual.

And finally, the bulk of this chapter details four *sequential* installation phases. We strongly recommend that you complete all steps in a phase, in order, before moving to the next phase. Your circumstances will determine which phases you need to do.

- **Phase One: Planning the Installation** -- sizing disks, naming file systems, listing devices, comparing DG/UX and foreign directory trees.
- **Phase Two: Loading the Primary Release** -- using **diskman**(1M) to load the primary release and create the OS logical disks and file systems.
- **Phase Three: Customizing the Primary Release** -- booting the starter system, loading software packages, building a custom kernel, setting a default boot path, creating other logical disks and file systems, adding user accounts, adding terminals, adding printers.
- **Phase Four: Adding OS Releases and Clients** -- adding secondary releases, adding diskless clients to a release, setting up diskless clients.

If you have a preloaded physical disk containing the basic software configured to default values, you may be able to skip all of Phase Two. Even if you have a preloaded disk, you'll still need to understand concepts, plan for your specific needs, and execute commands to create a usable system.

Important: Read all of Phase 1 before starting the installation.

Before You Begin

On the opposite page is a 22-step installation checklist composed of the major section titles of this chapter. This is your path from start to finish. We're assuming you have read Chapter 1 thoroughly and you are now familiar with the **sysadm** program, and you have a plan in mind for setting up a stand-alone host, an OS client, an OS server, or a combination of these.

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

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

Reading the checklist carefully will give you a sense of both the straightforwardness and the complexity of what you are about to do. Make a clear plan based on your changing future needs.

Checklist: Installing the DG/UX System

Phase 1: Planning the Installation

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

Phase 2: Loading the Primary Release

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

Phase 3: Customizing the Primary Release

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

Phase 4: Adding OS Releases and Clients

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

Our Example System

We'll be setting up a servnet consisting of one OS server and two diskless clients. The components of the system are:

- **Server**: One AViiON system named **sales**
- Two 322-Mbyte Ciprico ESDI disks
- One Systech asynchronous controller
- One Interphase Hawk ethernet network controller
- One SCSI tape drive with SCSI adapter
- One serial lineprinter
- **Diskless Client Workstations**: **dg1** (AViiON) and **sun1** (foreign)

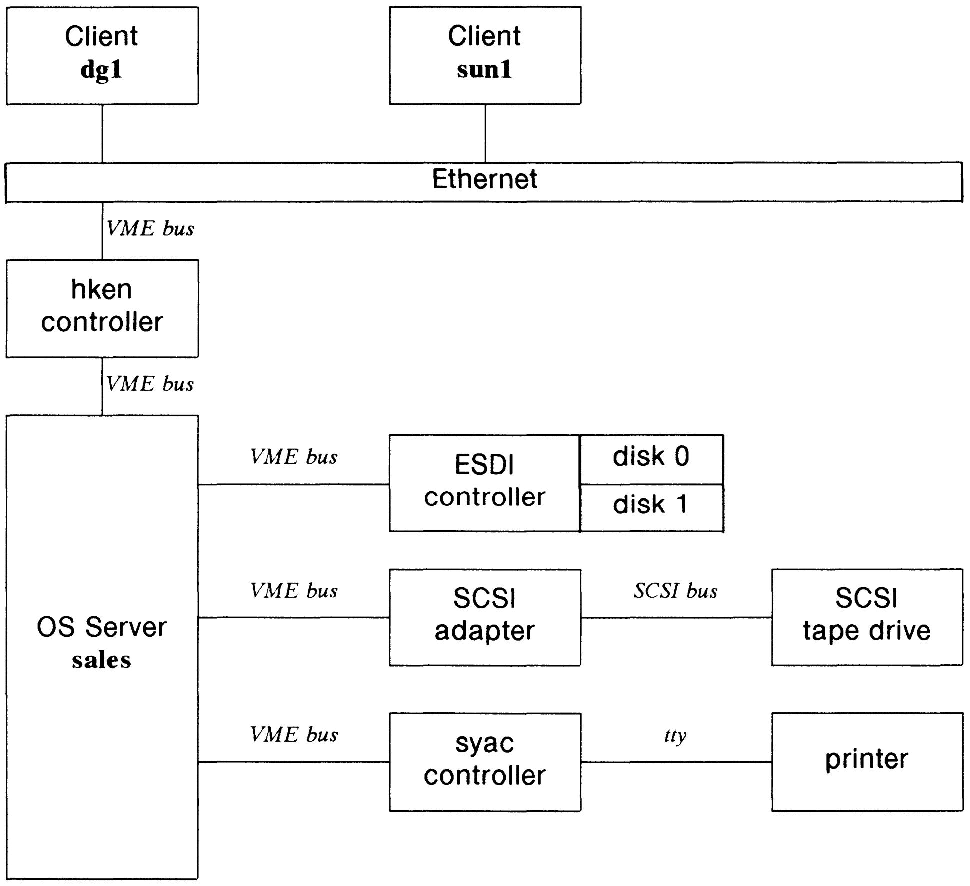


Figure 2-1 An Example Servnet

Device Specifications: General Background

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

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

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

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

Syntax

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

device_mnemonic [@device_code] ([parameters])

where the fields are as follows:

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

The following are valid device specifications:

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

Simplified Device Specifications

Most of the time you will not need to type in a long specification, such as **sd(cisc(1),2)**. A time when you will have to use this form is when interacting with the **diskman** program. But when you boot your system for the first time, a script runs called **chk.devlink**. This script reads the device files in **/dev** and sequentially maps each device to a number starting with 0 through n devices of a particular kind. When the mapping is complete, you can type in the number instead of the specification when you need to specify a device. For instance, 0 represents your first tape drive, 1 represents your second disk, and so on. Read your **/etc/devlinktab** file after booting to see how your devices are mapped. You can change the mappings in this file and your changes will take affect the next time you boot.

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

Phase One: Planning the Installation

Planning the system involves:

- Planning Resources and Using DG/UX Conventions
- Planning Disk Use for Releases
- Listing Devices on Your System
- Assembling Information for the Example System
- Understanding the DG/UX Directory Tree

Step 1: Planning Resources and Using DG/UX Conventions

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

hosts	Machine names should be such that you can easily associate them with their origins. This is especially valuable in an environment where many remote file systems are routinely mounted. For example, you could name machines according to owner, group, project, or department. Do not use the capitalized names *MY_HOST* or *PRIMARY*; these names are reserved by **sysadm** for managing releases.
releases	Release names are used by **sysadm** in the following form: *architecture_os_version* For example, the DG/UX release name is **88k_dgux_4.20**, which also happens to be the primary release for our example system.
logical disks	DG/UX loading facilties and internals require that system software is on logical disks named **root, usr,** and **swap**. Other logical disks, should be named such that they are easy to associate with their contents or origin. For example, on a host named **sales**, you might have a logical disk named **sales_accounts** to associate it with the specific machine. The file system on which **sales_accounts** might be mounted on the OS server would be **/sales/accounts**. Other hosts on the network would always know where the file system was physically located.
packages	This refers to packages such as the X Window System™ (X11). You can make logical disks for each package or put all packages in a single logical disk. For package X11, we'll create a logical disk named **usr_opt_X11**. This logical disk would be mounted on

/usr/opt/X11. Clients would mount this file system to access windowing software.

root directories — Client machines will receive their root directories by mounting a file system on the OS server during the netboot process. This file system has the form:

/srv/release/*release_name***/root/***client_name*

For example, a client host named **dg1** would mount the following on **/** and **/swap**:

/srv/release/PRIMARY/root/dg1 on **/**
/srv/swap/dg1 on **/swap**

local software — We suggest you use the directories **/local** (host-dependent)and **/usr/local** (release-dependent shared) for your site-specific shell scripts, programs, etc.

Step 2: Planning Disk Usage for the Release

Although you may have received a preloaded disk, you still have to create logical disks to complete your environment whether you're setting up as a stand-alone or a server machine. A preloaded disk means that some logical disks already exist on your physical disk when you start installation (**root, usr, usr_opt_X11, usr_opt_imagen,** and **swap**).

You will have to plan your physical disk use based on your specific needs. Our sample installation includes one OS server and two diskless client machines, **dg1** and **sun1**. Since we have a server machine, we'll be realistic and think ahead. How many more clients will we be serving in the future? For the sake of example, let's say within the next year we intend to have four Data General OS clients and two foreign OS clients.

The first thing we need to do is write down what we think our system will look like. We find out what our system will look like by answering questions such as: What logical disks will we need? How will we distribute them across disks? How much disk space do we have? How do we allocate space for diskless clients? Table 2-1 shows how we have planned our physical disks (PD), logical disks (LD), and file systems. Entries with stars might have come preloaded on your disk.

Table 2-1 A Logical Disk-File System Plan

Disk Type	PD Name	LD Name	Piece	Mounted File System Name
Ciprico ESDI	cied(0,0)	* swap	1	
		* root	1	/
		* usr	1	/usr
		* usr_opt_X11	1	/usr/opt/X11
		* usr_opt_imagen	1	/usr/opt/imagen
		sales_accounts	1	/sales/accounts
Ciprico ESDI	cied(0,1)	srv	1	/srv
		srv_dgux420	1	/srv/release/PRIMARY/root
		srv_sunos4	1	/srv/release/68k_sunos_4.0
		srv_swap	1	/srv/swap
		var_tmp	1	/var/tmp

System Logical Disks

We have set default sizes for the **root** and **usr** logical disks. A block is 512 bytes. The minimum **usr** block size will accomodate the DG/UX operating system, TCP/IP (10,000 blocks), and NFS (20,000 blocks). You may want to make your **usr** smaller if you don't intend to run TCP/IP and NFS.

Application Packages --

For other packages, such as X11, you can use one logical disk to contain all packages, or you can create a separate logical disk for each package. To minimize disk use, we'll make one logical disk to fit the size of each package.

Swap Space --

Swap logical disks differ from others in that they don't usually need to be file systems; they exist to ensure that memory is not in contention among processes waiting to run and processes ready to run. So that processes waiting to run do not take up memory space, they are written out to the swap space until they are ready to run. So we'll need a system **swap** logical disk for the server, and we'll need another for our OS clients (**srv_swap**). You'll notice that **srv_swap** is a file system so that our diskless clients can swap over the network. The amount of swap space needed is a function of a machine's physical memory. Multiply your physical memory by 1.5 to get the amount of swap space you need. The minimum swap space we recommend is 33,000 blocks. If you specify less, **diskman** returns you to the main menu.

Our main system logical disks are

root	* default: 40,000 blocks (single-piece logical disk)
usr	* default: 160,000 blocks (single-piece logical disk)
swap	* default: 30,000 blocks
usr_opt_X11	* default: 105,000 blocks

Client Logical Disks

Later, we'll be creating other logical disks to suit our needs as an OS server. We need space for client home directories, client swapping, for our window application package, root space for clients running the primary release, root and usr space for two foreign clients, and some general **tmp** space.

We want our primary OS clients to share the same kernel. In order for this to be possible, all primary clients' roots must be able to share the same logical disk space. So we must size the logical disk based on the total number of clients that will share it. Although we only have one primary client now, we're assuming that in the near future we'll be adding three more, thus we'll make a **srv_dgux420** logical disk of adequate size for four client root directories.

The logical disks we need are

srv	* Minimum: 2000 blocks (required for servers only)
srv_swap	* 128,000 blocks (four clients at 32,000 blocks each)
usr_opt_X11	* 105,000 blocks (the package size)
usr_opt_imagen	* 15,000 blocks (the package size)
srv_dgux420	* 160,000 blocks (primary release for 4 clients)
srv_sunos4	* 270,000 blocks (secondary release for 3 clients)
var_tmp	* 30,000 blocks

You should consider leaving part of your physical disk space free for adding future packages.

Calculating Primary and Secondary Logical Disk Space

Here is how we calculated the space we needed for our client release areas.

Primary Releases --

For diskless clients running the primary **88k_dgux_4.20** release, we'll create a **srv_dgux420** logical disk according to the following formula:

`No. clients * (client root size)=` **srv_dgux420**

```
4 * (40,000) = 160,000 blocks
```

Secondary Releases --

Here's how we calculate a secondary release space for our two foreign clients. Note that we add an extra 150,000 blocks (or whatever is necessary) for the foreign system's **usr** space.

`total client root sizes + usr size =` **srv_sunos4**

```
120,000 + 150,000 = 270,000 blocks
```

Be sure to check the documentation for foreign systems for the correct amount of space to allocate for **/** and **/usr**.

Note that we created a **var_tmp** logical disk. Since **/var** is part of the **root** logical disk and we expect our system to make substantial use of the **/var/tmp** area, we decided to set up a separate logical disk. This protects the rest of the root file system from filling up if **/var/tmp** fills up. We could have done the same thing with **/var/mail** or any other subdirectory of **/var** that doesn't contain files shipped with the

DG/UX release.

Home Directories

Our diskless clients will need home login directories. For this purpose, we'll create the **sales_accounts** logical disk. We have about 283,000 blocks left on this disk. Let's assume that 200,000 blocks will be enough for this logical disk. We'll leave the rest of the disk as free space for future needs.

Toggling Between Primary Releases

Most servers will run a single primary operating system. If you should want to set up your server so that it can run more than one primary operating system, you will have to plan your disk usage differently than we did in Figure 2-2. Our plan sets up our server to run only one primary operating system. We would have to reload a new primary system in order to run another one. If you want to have the ability to shutdown one primary OS and then reboot another primary OS, then you need to set up three additional logical disks for the second primary OS. For example, if you wanted to toggle between DG/UX 4.10 and DG/UX 4.20, you might have additional logical disks as follows:

dg_410_usr	To mount as a second **usr**.
dg_410_serv_root	To mount as the server's **/** file system.
dg_410_client_root	To contain the roots for your clients.

The Completed Disk Layout

Figure 2-2 shows how we have allocated our two physical disks for our example system. The Internal Area contains the Primary System Area (PSA), bootstrap programs, the Logical Disk Piece Table, Bad Block Tables, and the Bad Block Remap Area.

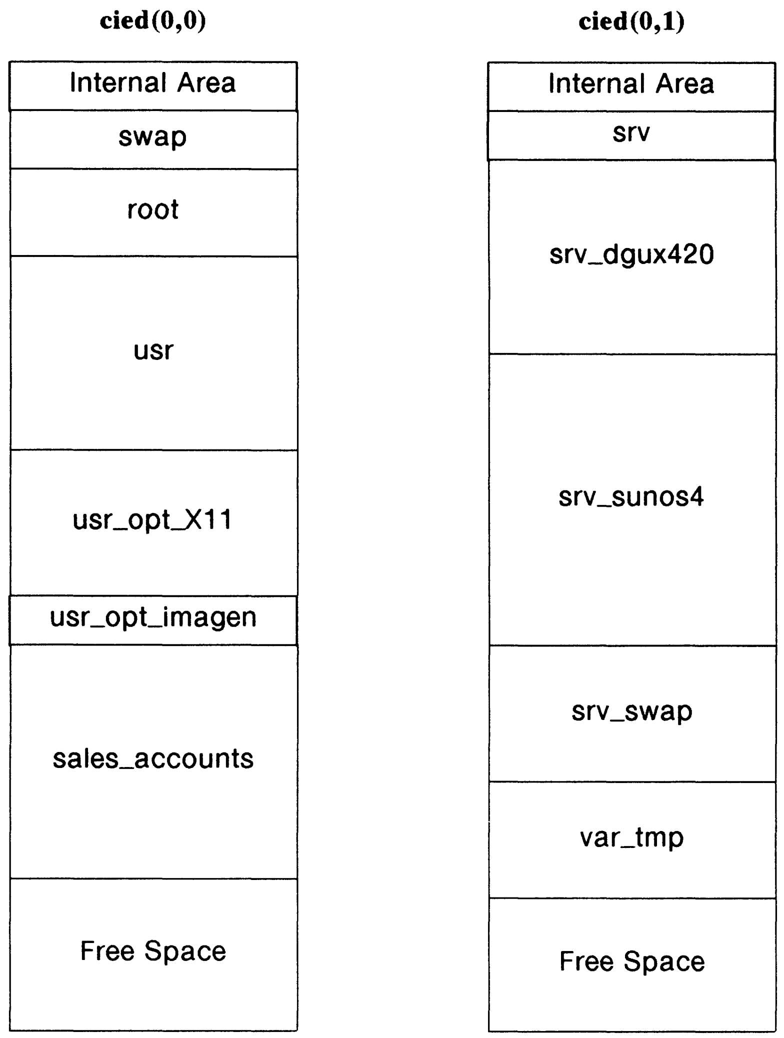


Figure 2-2 Disk Layout for the Example Servnet

Notice that we have most of the logical disks related to our servnet on one disk. A multiuser or stand-alone system might have a similar layout as our first disk.

Step 3: Listing the Devices on Your System

Table 2-2 shows the device information pertinent to our example system. See *Using the AViiON System Control Monitor (SCM)* for detailed information on setting device information for your particular system.

Table 2-2 Device Information for Our Example System

Device	Specification	Device No.	SCSI ID No.
First disk controller (boot disk)	cied(0,0)	18	NA
Second disk controller	cied(0,1)	19	NA
Network controller	hken()	15	NA
asynchronous controller	syac()	60	NA
Tape drive	st(cisc(),4)	NA	4

We'll see the information in this table again when we format logical disks and edit the configuration file before building a new kernel later in this chapter. You might make a similar table for your own use.

Note that you'll never need to use the device numbers from this table, but you will use the SCSI ID number when specifying your tape drive to the **diskman** program. These entries are provided here to show you what the default device numbers are. If, for example, you use up all disk defaults (by adding new disk devices to your system), then you'll need to specify a non-standard base address for each new device. For example, if you added a third ESDI disk controller, you would have to choose a base address for it. Your system file entry for the third ESDI disk controller might be:

```
    cird@80(fffff50,0)
```

The `cird` notation covers all **cied** and **cimd** devices. See Appendix A for more information on devices and base addresses.

Step 4: Assembling Information for the Example System

Knowing that we'll have three machines in a servnet, we've listed the following information so that we'll have it handy when needed for other parts of the installation. See *Installing and Managing DG/UX™ TCP/IP* if you need information on obtaining an Internet Protocol (IP) address. A machine's Ethernet address is displayed at each powerup.

Table 2-3 Assembled Information for the Example Servnet

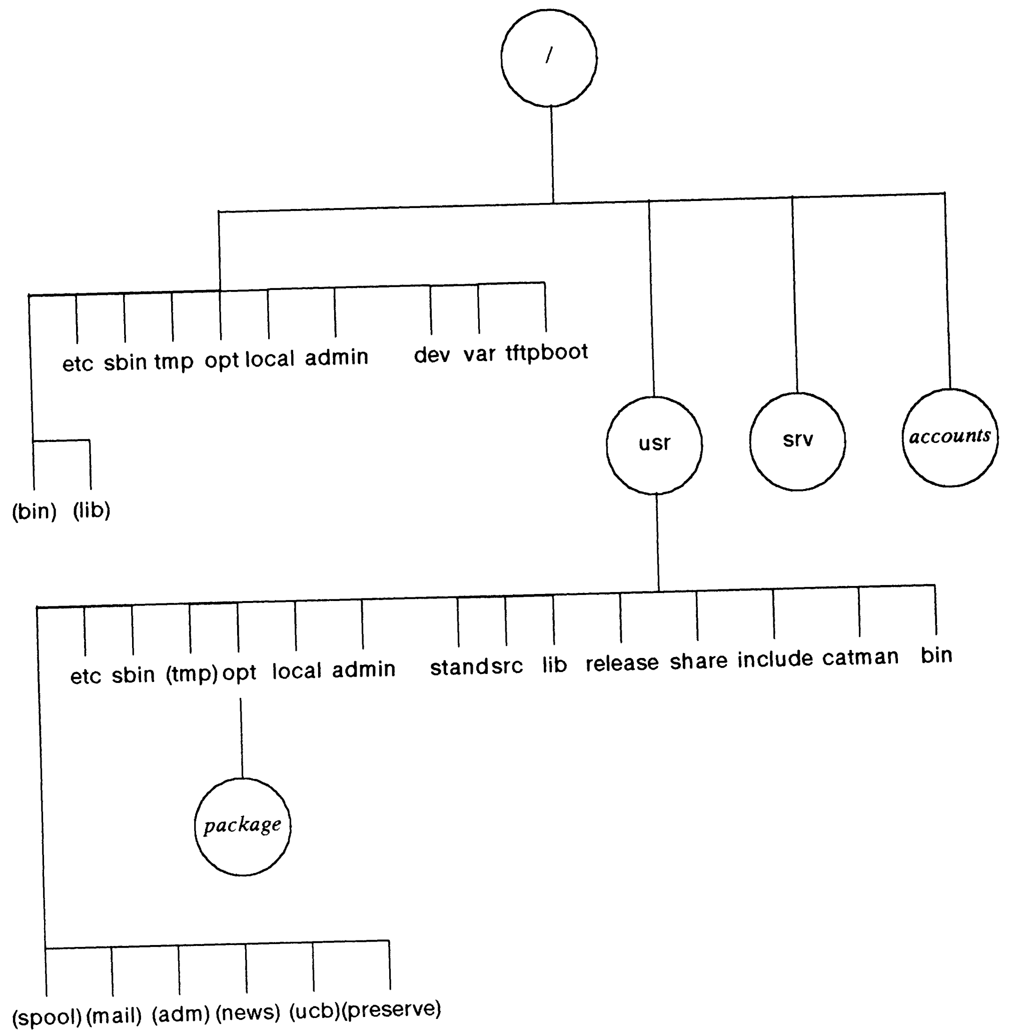
Host	IP address	Ethernet address	Release
sales	128.223.2.1	04:00:1c:00:2a:12	primary
dg1	128.223.2.2	08:00:1b:00:a0:17	primary
sun1	128.223.2.3	08:00:20:00:a7:5d	68k_sunos_4.0

The empty table below is provided for your host information.

Host	IP address	Ethernet address	Release

Step 5: Understanding the DG/UX Directory Tree

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


/
etc sbin tmp opt local admin dev var tftpboot
(bin) (lib)
usr
srv
accounts
etc sbin (tmp) opt local admin standsrc lib release share include catman bin
package
(spool) (mail) (adm) (news) (ucb) (preserve)

Circles are logical disks (file systems) mounted on directories. In our example system:

accounts is logical disk **sales_accounts**
package is the mount point for **usr_opt_X11**

Names in parentheses are symbolic links to directories.

Figure 2-3 An Example DG/UX Directory Tree

The / Directory Tree

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

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

The /srv Directory Tree

Our sample system is made up of one OS server, one native client, and one foreign client. Remember that the steps we go through in setting up our OS server are the same ones you will do if you're setting up a stand-alone machine instead. Our **/srv** looks like this:

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

The primary release, PRIMARY, has links to the server's **root** and **usr** logical disks. Having a **/srv** directory allows the **sysadm** program to manage releases and clients regardless of which DG/UX release the server is running.

The /usr Directory Tree

The file system mounted on **/usr** contains architecture-dependent and read only, shareable files. Directories where writes to host-dependent files must occur are indicated as symbolic links below.

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

The **/var/spool** directory replaces the traditional DG/UX **/usr/spool** directory. Files that are host dependent and change size dynamically (print and mail queues, log files, etc.) are located in **/var**. Symbolic links preserve the traditional directory tree structure of **/usr**. Depending on your usage, you may want to consider making logical disks for those directories with stars and mounting them under **/var**.

File System Mapping: DG/UX to Foreign

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

```
/srv
  |
  |--- release
  |      |--- PRIMARY (88k_dgux_420)
  |      |        |
  |      |        |--- usr   --> /usr
  |      |        |--- root
  |      |              |--- MY_HOST   --> / (root)
  |      |              |--- dgl
  |      |
  |      |--- 68k_sunos_4
  |              |--- usr
  |              |--- root
  |-- share      |--- sunl
  |--- swap
  |     |
  |     |--- dgl
  |     |--- sunl
  |
  |--- admin            (sysadm data files)
```

A foreign system **/export** structure:

```
           |--- root
           |     |--- server  (primary release: symbolic link to /)
/export---|     |--- dgl
           |     |--- sunl
           |
           | --- exec
           |      |--- 88k_dgux_420
           |      |--- 68k_sunos_4
           |
           |--- swap
           |     |--- server
           |     |--- dgl
           |     |--- sunl
           |
           |--- share
```

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

- Each client can be attached to multiple releases and maintain a separate root for each release. This way, clients can choose among releases they wish to boot.
- The **srv** file system contains data files used to maintain clients and releases which are not associated with a particular root on the server. If the server boots another release, it won't lose information about clients and releases.

Phase Two: Loading the Primary Release

If you have a preloaded disk, this phase may have already been done for you. If so, you may skip ahead to Phase 3, "Customizing the Primary Release."

By primary release we mean the total contents of the DG/UX system package you purchased. The package we're using for our example contains the DG/UX operating system, TCP/IP, ONC/NFS and some other products. We'll load and set up these products later in Steps 13 and 14.

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

The steps in this phase are

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

Step 6: Booting diskman from Tape

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

We are booting on an AViiON server. If you are booting an AViiON workstation, your command line will differ depending on your tape device. For instance, you might use **st(insc(),4)**.

Mount your installation tape and type the following after the SCM> prompt. Our entry is the specification of the tape drive for the example system:

```
SCM>  b st(cisc(),4) ↵
```

This command loads the release tape file 1 into memory and comes up in a device menu for diskman and queries you about device names. If you are on a server, respond with **duart()**. If you are on a workstation, respond with **kbd()** and then **grfx()**. After you complete these steps, **diskman** begins execution. Once you are in **diskman**, you will need to know the proper names for the devices on your server or workstation. See Appendix A for more information on device names.

The following table shows you how to use the **diskman** menus.

Table 2-4 Using Diskman Menus

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

When the system has loaded **diskman**, you will see the opening menu:

```
               Diskman Main Menu

1. Physical Disk Management Menu

2. Logical Disk Management Menu

3. File System Management Menu

4. Initial Installation Menu

5. Update Installation Menu

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

Enter Choice: [4]
```

Enter **4** to begin initial installation. This option will lead you through all the steps necessary to get your system running in single-user mode.

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

```
             Initial Installation Menu

1. Initialize Physical Disks

2. Create the Root Logical Disk and File System

3. Create the Swap Logical Disk

4. Create the /usr Logical Disk and File System

5. Load the Root File System

6. Load the /usr File System

7. All Installation Steps

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

Enter Choice: [7]
```

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

Step 7: Initializing Physical Disks with diskman

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

Surface analysis is very time consuming at this point because you can do only one physical disk at a time. For now, you must run surface analysis (individually) on the physical disk or disks that will contain the **root, swap,** and **usr** logical disks.

After you choose number 7, the system responds as follows. The responses in **bold** type are correct for the example system described earlier. We begin by formatting the physical disk that will contain the **root, swap,** and **usr** logical disks:

```
1. Initialize Physical Disks

Do you want to run this step? [y] ↵

Enter the Physical Disk specification in DG/UX
common format: cied(0,0)

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

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

     Disk Types

1. 6442   ESDI   322Mb
2. 6555   ESDI   648Mb
3. 6491   SCSI   322Mb
4. 6554   SCSI   622Mb
5. 6541   SMD    1066Mb
6. 6355   SCSI   179Mb
7. None of the Above.

What type of disk do you have? 1 ↵
Disk label has been installed.
```

```
    Perform Hardware Formatting on a Physical Disk

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

  WARNING: This operation will DESTROY any data on the Physical
  Disk cied(0,0).

    Do you want to continue? [y] ↵
    Disk cied(0,0) has been formatted.


    Create DG/UX System Areas on a Physical Disk

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

  WARNING: This operation will DESTROY any data on the Physical
  Disk cied(0,0).

    Do you want to continue? [y] ↵

    The Physical Disk cied(0,0) is 628906 blocks in size.
    Enter the number of blocks to allocate for the remap
    area: [189] ↵


    Install Bootstraps on a Physical Disk

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

    Installed Bootstraps on the Physical Disk cied(0,0).


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

If Your Disk Has a Built-in Bad Block Table

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

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

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

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

Even if your disk has a built-in bad block table, we recommend that you run surface analysis on it anyway to be sure that there are no problems. We recommend that you press New Line to enter the default choice, which runs all the test patterns. If you enter no, **diskman** will run only one test pattern. The process takes about 20 minutes per 100 Mbytes, depending on your physical disk model and CPU. An average is about an hour per physical disk.

For our example system, you would see the following progress message about every five minutes:

```
Beginning Surface Analysis...
Surface Analysis 9 percent complete; finished in 50 minutes
Surface Analysis 18 percent complete; finished in 45 minutes
```

As another five-minute increment of surface analysis is completed, you will see another progress message. After all patterns have run, the bad block table is written to the disk. You'll see the following message.

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

If you need to format another physical disk, type **y**. The program for intializing physical disks will repeat from the beginning.

For the example system, we need to initialize another physical disk, **cied(0,1)**. Run the initialization process as before. When you're finished initializing your disks, proceed to the next step: Creating System Logical Disks and File Systems.

Step 8: Creating System Logical Disks and File Systems

In this section, we'll create the system logical disks, **root, usr**, and **swap**.

Creating the Root Logical Disk and File System

Here, **diskman** creates a **root** logical disk and makes the **/** (root) file system. The dialogue begins as follows:

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

Do you want to run this step? [y] ↵

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

We used the default name (root). This is the node name that the operating system will use in **/dev/dsk** and **/dev/rdsk**.

```
Enter the Physical Disk specification in DG/UX common
format: [cied(0,0)]  ↵

Do you want to display the layout of this
Physical Disk? [n] ↵

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

The default begins at the first available space on the disk.

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

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

```
The Logical Disk "root" has been created.
Making a file system on the Logical Disk...
Made a file system on the Logical Disk "root".
```

Creating the Swap Logical Disk

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

```
3. Create the Swap Logical Disk
```

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

Enter Logical Disk Name:  [swap] ↵

Enter the Physical Disk specification in DG/UX common
format: [cied(0,0)]  ↵
```

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

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

In our example, the default calculated by **diskman** begins at the first block after the end of the **root** logical disk.

```
Enter the size in blocks of the Logical
Disk Piece: [50000]  ↵

The Logical Disk "swap" has been created.
```

We started with 322 MB (628906 blocks). We still have about 538,906 blocks left on this disk.

Creating the usr Logical Disk and File System

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

Do you want to run this step? [y] ↵

Enter the Logical Disk Name: [usr] ↵

Logical Disk Piece 1:
Enter Physical Disk specification in DG/UX common
format: cied(0,0)

Do you want to display the layout of this
Physical Disk? [n]  ↵

Enter the Physical Disk Address of the starting block
of Logical Disk Piece 1: [default]  ↵
Enter the size in blocks of Logical Disk
Piece 1: [180000] ↵
```

The **diskman** program loops back for information on up to seven more logical disk pieces. We complete the logical disk pieces for our example as follows.

```
Do you want to specify any more Logical
```

```
Disk Pieces? [n] ↲

The Logical Disk "usr" has been created.
Made a file system on Logical Disk "usr".

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

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

In this section, we load the **/** and **/usr** file systems.

Loading the / File System

In step 5 of the initial installation, **diskman** loads the **/** file system's files from the release tape to the location you created for it. These files include a starter system that contains a minimum of information on disk and tape devices.

```
5. Load the Root File System

Do you want to run this step? [y] ↲
Do you want to see the names of the files being loaded? [y] ↲

Enter the Logical Disk Name: [root] ↲

Enter the tape drive specification in DG/UX common
format: st(cisc(),4) ↲
```

You must enter the name of the specific tape drive unit that you will use to load files from tape.

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

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

Make sure that your tape is mounted and online, then press New Line. The system now displays the names of the files as they are loaded (if you answered yes to the above question).

```
Loading...
.
.
.
n blocks

The Root File System has been loaded.

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

Loading the /usr File System

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

```
6. Load the /usr File System

Do you want to run this step? [y] ↵

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

Enter the Logical Disk Name:  [usr] ↵

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

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

Press New Line when ready to continue... ↵

Loading...
  .
  .
  .
```

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

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

Press New Line when ready to continue... ↵

Your starter system has been installed.
You can now boot the DG/UX starter kernel from disk.
```

If you have additional update software to add, stay in the **diskman** program and go to the next step in this phase, "Updating System Software." Otherwise, you can now boot your starter kernel while you're running **diskman**.

```
Do you want to boot the starter kernel? [y] ↵
```

Now we'll go to "Phase Three: Customizing the Primary Release."

Step 10: Updating System Software

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

Since your are already in the stand-alone **diskman** program, return to the Main Menu and select number 5, "Update Installation Menu." Your system will display the following:

```
                Update Installation Menu

1. Update the Root File System

2. Update the /usr File System

3. All Update Steps

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

Enter Choice: [3]
```

Mount your update tape, then select number 3. The system displays:

```
1. Update the Root File System

Do you want to run this step? [y] ↵

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

Enter the Logical Disk Name: root ↵

Enter the tape drive specification in DG/UX common
format: st(cisc(),4) ↵
```

You must enter the name of the specific tape drive unit that you will use to load files from tape. Make sure the tape is online and at BOT. You will be prompted for each tape.

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

The system now displays the names of the files as they are loaded (if you answered yes to the above question).

```
Loading...
```

```
.
n blocks

The Root File System has been updated.

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

Pressing New Line takes you automatically to the next step in the Update Installation Menu.

Updating the /usr File System

In this step, **diskman** loads the **/usr** file system from the update tape.

```
2. Update the /usr File System

Do you want to run this step? [y] ↵
Do you want to see the names of the files being loaded? [y] ↵

Enter the Logical Disk Name: usr ↵

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

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

Loading...
  .
  .
```

You will be prompted to load each tape. When you have loaded the last tape, you will see these messages:

```
The /usr file system has been updated.
Press New Line when ready to continue...↵

Your system has been updated.
You can now boot the DG/UX starter kernel from disk.

Do you want to boot the starter kernel? [y]
```

Phase Three: Customizing the Primary Release

The steps in this phase are

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

Step 11: Booting the Starter Kernel

The last thing we did in Phase Two was boot our starter system using **diskman**. If you have a preloaded disk which allowed you to skip Phase Two, then you need to boot the starter system from the SCM> command line.

Booting an OS Server or Multiuser System

We're booting an OS server with ESDI disks. If we needed to boot the starter system from the SCM> command line, we would type

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

where *m* is the disk controller number and *n* is the disk unit number.

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

To boot an OS server with a SCSI disk, you would use a command line like this:

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

where *m* is the disk controller number and *n* is the disk unit number (probably **0** if a disk device or **4** if a tape device).

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

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

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

where *m* is the disk controller number and *n* is the disk unit number.

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

Booting a Standalone Workstation

If you're booting an AViiON workstation, your specification will be different; it will reflect the devices specific to your hardware. For instance, you might boot with the command

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

where *m* is the disk controller number and *n* is the disk unit number.

The first thing we have to do as the starter system comes up is enter the names of a minimum number of devices. The following table shows the devices for AViiON systems and workstations.

Table 2-5 Starter System Devices

AViiON system	AViiON workstation
cird()	sd(insc(),0)
st(cisc(),4)	st(insc(),0)
duart()	duart()
	kbd()
	grfx()

After you have entered your device names, press the New Line key to the last `Device Name?` prompt. For more complete information about device names, see Appendix A.

The mnemonic **cird** covers **cied** and **cimd** devices.

The system displays the following:

```
                DG/UX Starter System

Enter the names of the devices you will use in
DG/UX Common Specification Format. Enter just newline
when done.

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

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


** root:  No check necessary for root

INIT: Cannot open /etc/TIMEZONE. Environment not initialized.
INIT: /etc/inittab file created from prototype file
INIT: Checking and mounting /usr...
INIT: /usr is now mounted

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

#
```

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

Setting Up DG/UX: Initial Configuration

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

To change run levels from single-user level S to administrative level 1, type:

```
# init 1

chk.fsck:

chk.date:
   Current date/time: Wed Jun 10 08:15 EDT 1989
   Is the current date, time, and TIMEZONE  correct? [n]: y ↵

Setting up package: dgux
```

```
Initializing system database files from .proto files:
```

Messages about initializing prototype files will differ depending upon your configuration.

```
initialize /etc/passwd
initialize /etc/group
initialize /etc/dgux.params
initialize /etc/dgux.rclinktab
initialize /etc/dumpdates
initialize /etc/dumptab
initialize /etc/gettydefs
initialize /etc/login.csh
initialize /etc/motd
initialize /etc/profile
initialize /etc/stdlogin
initialize /etc/stdprofile
initialize /etc/syslog.conf
initialize /etc/TIMEZONE
initialize /etc/wtmp
initialize /etc/admin/.profile
initialize /etc/sysadm/architecture
initialize /etc/sysadm/dumpcycle
initialize /etc/sysadm/mt
initialize /etc/sysadm/timezone
initialize /etc/sysadm/tty
initialize /etc/sysadm/user
initialize /etc/sysadm/uucp

Setting up the rc*.d directory links.
Cleaning links in /etc/rc#.d directories.
.............................................
Making rc script links to source directory.
.............................................
Initializing system database file from .proto files:
initialize /usr/lib/acct/holidays
initialize /usr/lib/uucp/Devices
initialize /usr/lib/uucp/Dialcodes
initialize /usr/lib/uucp/Dialers
initialize /usr/lib/uucp/Permissions
initialize /usr/lib/uucp/Sysfiles
initialize /usr/lib/uucp/Systems
initialize /usr/lib/uucp/remote.unknown
initialize /usr/lib/uucp/Poll

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

** WARNING: These local accounts have NO password.
```

```
    root::0:1:root:Special Admin Login/:/sbin/sh
    sysadm::1:sysadm:Regular Admin Login/admin:/sbin/sh

 chk.devlink:
    Add short names (for device nodes) to /etc/devlinktab
    /dev/rmt        0         st(insc@EFFF8A000),4,0)
    /dev/pdsk       0         cied@18(FFFFEF00)
    /dev/rpdsk      0         cied@18(FFFFEF00)
    /dev/pdsk       1         cied@19(FFFFF100)
    /dev/rpdsk      1         cied@19(FFFFF100)

    Link short names for /dev device nodes:
    /dev/rmt/0   -->  /dev/rmt/st(insc@E(FFF8A000),4,0)
    /dev/rmt/0n  -->  /dev/rmt/st(insc@E(FFF8A000),4,0)n
    /dev/pdsk/0  -->  /dev/pdsk/cied@18(FFFFEF00)
    /dev/rpdsk/0 -->  /dev/rpdsk/cied@18(FFFFEF00)
    /dev/pdsk/1  -->  /dev/pdsk/cied@19(FFFFF100)
    /dev/rpdsk/1 -->  /dev/rpdsk/cied@19(FFFFF100)


    Executing the /etc/rc1.d scripts.

    Starting syacs: /usr/lib/syac/syacload -a
    Starting rc.update: update
    Starting rc.localfs: mount  -at  dg/ux

    The following file systems are now mounted:

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


    Starting rc.setup:
      Check for packages that haven't been set up.

    Press <RETURN> to display prompt.

    no-node
    DG/UX Release 4.20
    login: sysadm ⮐
```

Note that above we typed the **sysadm** login, instead of the traditional **root** login. This logs you in to **/admin** and gives you all the superuser privileges of the **root** login. If you keep **.profile** and other personal files in **/admin**, this ensures that **/** (root) remains a clean, protected directory that you can always boot.

Step 12: Creating Other Logical Disks and File Systems

This section shows how we create remaining logical disks on our two physical disks. We'll refer to Figure 2-2 where we planned the contents of both disks. Note that you may not have to make the **usr_opt_***PKG* logical disks listed below; they may have come preloaded on your system. The sizes (in 512-byte blocks) of the logical disks we will create are:

1st Physical Disk ---- **cied(0,0)**

usr_opt_X11	105,000 blocks (may be preloaded)
usr_opt_imagen	15,000 blocks (may be preloaded)
sales_accounts	200,000 blocks
Free Space	about 35,000 blocks

2nd Physical Disk ---- **cied(0,1)**

srv	2000 blocks
srv_dgux420	160,000 blocks
srv_sunos4	270,000 blocks
srv_swap	128,000 blocks
var_tmp	30,000 blocks
Free Space	about 50,000 blocks

Let's work on physical disk **cied(0,0)**. To begin, return to the Logical Disk Management Menu. Select number 1, Create a Logical Disk.

Using our plan, we need to create the rest of our logical disks. You know how to use **diskman**, and you should have a plan of your own by now, so go ahead and make the rest of your logical disks. Use **sysadm diskmgmt**.

Adding File Systems to /etc/fstab with sysadm

Now that the file systems are created, we need to make them accessible. We do this with the **sysadm addfsys** command. For instance, after we create the **sales_accounts** logical disk and file system, we'll need to mount it on **/sales/accounts**. We mount as follows:

```
# sysadm addfsys ↵

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

Mount Directory Name?  /sales/accounts ↵
Is this a local file system?  [yes]  ↵
Writeable?  [yes]  ↵
Dump Cycle? [d] ↵
fsck Pass? [1] ↵
Export the file system? [no] y ↵
The entry for /sales/accounts has been added.
The directory /sales/accounts does not exist.
Create /sales/accounts? [yes]  ↵
Mount the file system?  [yes]  ↵
The file system has been mounted.
```

Invoke the **addfsys** command again to mount the rest of your file systems. Diskless clients will use this same procedure to gain access to the file systems on the server's disk. See Chapter 8 for more information on accessing file systems. Refer to Table 2-1 where we planned the mount point directories for our file systems. You should have a similar plan for your file systems.

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

Step 13: Loading Software Packages with sysadm

The **sysadm loadpackage** function allows you to load all software packages at once that are on a single tape. TCP/IP, NFS, and the X Window System™ are among the packages on our tape Each package name will be displayed; we'll select the ones we want to load.

Note that among our packages is one window applications package which will be loaded in the **/usr/opt/X11** file system. The **/usr/opt/X11** file system must be mounted before a package can be loaded into it. With this mount done, the package will automatically be loaded in the correct location. Note that **loadpackage** loads files into a release area relative to the load point specified in the release's tape table of contents.

Before you use **loadpackage**, you must first run **sysadm makesrv** to create the **/srv** directory tree. Then begin loading packages.

We'll begin by typing:

```
# sysadm loadpackage ↵

Release Area? [PRIMARY]  ↵
Tape Drive? [0] ↵
Is the Tape Mounted and Ready? [yes] ↵

Load Package dtk? [yes] ↵
Load Package gcc? [yes] ↵
Load Package onc-nfs? [yes] ↵
Load Package tcpip? [yes] ↵
Load package X11? [yes] ↵

List file names while loading? [yes] no ↵

Mount Volume 1.
Is the tape mounted and ready? [yes] ↵
loadpackage is finished.

#
```

Loading the above products took about 35 minutes.

Step 14: Setting Up Software Packages with sysadm

Different packages will require different kinds of set up. See the release notice for each product you load. In this step, we'll set up TCP/IP, build a new kernel, then set up NFS and YP. In setting up a package with **sysadm setuppackage**, you answer queries or supply parameters or data needed by a given package. Remember to run **sysadm makesrv** before setting up software packages.

See *Installing and Managing DG/UX™ TCP/IP* for detailed information on the TCP/IP product. We provide basic information concerning our example system in this section. To install TCP/IP, you'll need to know similar information about your system before you begin. To set up the OS server, **sales**, for our example system, our TCP/IP information follows:

host name	The OS server is named sales.
Internet address	The Internet address for sales is 128.223.2.1 .
network name	sales is on a network named skyhook.
broadcast address type	The type is all ones. (BSD 4.3)
controller type	The controller type is Hawk Ethernet, hken. For a workstation, it would be inen.
controller name	The controller name is hken0.
test host	For testing, you'll need the name of one running remote host and its Internet address.

TCP/IP

Since we have assembled the information we'll need, all we have to do is invoke the TCP/IP setup scripts via the **setuppackage** function. When we finish TCP/IP, we'll build a new kernel, then move on to NFS and the Yellow Pages. For now, let's assemble the information we need for NFS and YP.

NFS

See *Managing NFS and Its Facilities on the DG/UX™ System* for basic information and concepts. This section assumes that you are familiar with NFS and provides only minimal information.

NFS planning consists of compiling a list of remote hosts, remote host managers, and remote host file systems that are of interest to you. The parameters for NFS are in **/etc/nfs.params**. You can read this file for instructions on entries you may want to modify. A proto file will be initialized if you choose not to modify this file.

Yellow Pages (YP)

Setting up the Yellow Pages (YP) requires a thorough understanding of how the process works. This section only shows the specific information we need for our YP.

Below, we'll list the YP information for our sample system:

host name	sales
YP class	Of master, server, or client, we'll be the YP master.
master host	sales is the host that keeps the master YP database
network address	128.223.2.1
YP domain name	top.domain

Setting Up TCP/IP

Now that we've listed the information we need, let's invoke the command to begin setting up the first package:

```
# sysadm setuppackage ⮐

Release Name? [PRIMARY] ⮐

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

tcpip   onc-nfs   yp   X11
```

```
Package Name? [all] tcpip ↵
Processing setup scripts for package tcpip.
Set up package tcpip in usr? [yes] ↵

There is a naming convention conflict between the restricted
shell and the remote shell.  In revisions of DG/UX before 4.00
the restricted shell command was named restsh and the remote
shell command was named rsh.  Therefore, we have renamed the
remote shell command to be remsh.  You will be asked if you
prefer the old naming conventions.  If you choose the default,
the restricted shell will be installed as /usr/bin/restsh and
the remote shell as /usr/bin/rsh.

If you want SVID compliance, say NO.
If you want to keep the old naming conventions, pick the default.
Do you prefer the old naming conventions? [y]

Restricted shell is named /usr/bin/restsh.
Remote shell is named /usr/bin/rsh.

Remote Commands Installation Complete
Press NEWLINE when ready to continue...  ↵

Executing /usr/bin/newaliases..please wait

Sendmail Aliases Setup Complete
Press NEWLINE when ready to continue...  ↵

Setup Package tcpip in MY_HOST root? [yes] ↵

         Setting up package: tcpip

Creating links for initialization scripts...please wait

File: /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.rclinktab
has been created from prototype file.

Making rc script links to source directory.
.........................................................

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts has been
created from prototype file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/networks has been
created from prototype file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/services has been
created from prototype file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/protocols has been
created from prototype file.
```

```
File: /srv/release/PRIMARY/root/MY_HOST/etc/inetd.conf has been
created from prototype file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/ethers has been
created from prototype file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params has been
created from prototype file.

Press NEWLINE when ready to continue... ↵


Do you want support for loop interface? [y] ↵

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files...Please Wait.


NOTE: Any entries encountered containing conflicting information
will be deleted from the offending file.

The following lines have been removed from file
/srv/release/PRIMARY/root/MY_HOST/etc/hosts:

--Begin Remove List--
internet_address    localhost
--End Remove List

The entry "internet_address   localhost" has been added
to /srv/release/PRIMARY/root/MY_HOST/etc/hosts.

Updating "/srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params"

IMPORTANT NOTE: You MUST have a loop entry specified in your
system configuration file.  Consult the system(4) man page for
more information.

Local Loopback Environment Installation Complete
Press NEWLINE when ready to continue...

The following queries refer to the host being installed:

Enter Host Internet Address: 128.223.2.1 ↵
Enter Host Name: sales ↵
Enter Network Name: skyhook ↵
Is skyhook a subnetted network? [n] y ↵
Enter the network mask: 0xffffff00 ↵
Calculating the network address...Please Wait.

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files.
```

```
The entry "128.223.2.1   sales" has been added to
/srv/release/PRIMARY/root/MY_HOST/etc/hosts.

The entry "128.223.2   skyhook" has been added to
/srv/release/PRIMARY/root/MY_HOST/etc/networks.

Enter controller device name: hken0 ↵

There are two variations of Broadcast addresses.  A BSD 4.2
compatible broadcast address has a host portion of all zeros.
A BSD 4.3 compatible broadcast address has a host portion of
all ones.

Calculating network portion of broadcast address...please wait

Do you want the host portion of the broadcast address to be
all ones? [yes] ↵

Calculating broadcast address...please wait

Updating /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params.

IMPORTANT NOTE: You MUST have a "hken" entry specified in your
system configuration file.  Consult the system(4) man page for
more information.

Local Environment Installation Complete.
Press NEWLINE when ready to continue... ↵

Would you like to add a remote host entry? [y] ↵
The following refers to other hosts on this network:

Enter Host Internet Address: An existing host address
Enter Host name: An existing host name
The entry has been added to /etc/hosts.
Do you want to add another remote host entry? [n] ↵

Remote Environment Installation Complete
Press NEWLINE when ready to continue... ↵

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/protocols file? [n] ↵

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/services file? [n] ↵

Network Environment Installation Complete.
Press NEWLINE when ready to continue...

Enter FTP login directory
```

```
[/srv/release/PRIMARY/root/MY_HOST/var/ftp]: ↵

Modifying ftp password entry in
/srv/release/PRIMARY/root/MY_HOST/etc/passwd.

Directory: /srv/release/PRIMARY/root/MY_HOST/var/ftp
has been created.

Directory: /srv/release/PRIMARY/root/MY_HOST/var/ftp/bin
has been created.

Directory: /srv/release/PRIMARY/root/MY_HOST/var/ftp/etc
has been created.

File: /usr/bin/ls has been copied to
/srv/release/PRIMARY/root/MY_HOST/var/ftp/bin/ls.

File: /usr/bin/pwd has been copied to
/srv/release/PRIMARY/root/MY_HOST/var/ftp/bin/pwd.

File: /etc/group has been copied to
/srv/release/PRIMARY/root/MY_HOST/var/ftp/etc/group.

FTP Installation Complete.
Press NEWLINE when ready to continue...

WARNING: The following query may produce a security breach in your
system.  An entry in the hosts.equiv file allows a user from the
specified remote host having the same name to remotely login to
your host WITHOUT having to enter a password.  Caution should be
exercised in adding entries to this file.

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv has been
created from the prototype file.
Do you wish to add a host to the /etc/hosts.equiv file? [n] ↵

Remote Commands Installation Complete.
Press NEWLINE when ready to continue...


Do you wish to customize ruleset 0? [n] ↵
Modifying mail passwd entry in
/srv/release/PRIMARY/root/MY_HOST/etc/passwd.

Do you wish to use sendmail as the mailx router? [y] ↵

The directory /srv/release/PRIMARY/root/MY_HOST/var/mailx
has been created.
The file /srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc
has been created.
```

```
The entry "set sendmail = /usr/lib/sendmail" has been
added to /srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc

File: /usr/etc/aliases has been created from the proto file.
Do you want to edit the /usr/etc/aliases file? [n] ⤶

Sendmail Installation Complete


#
```

The TCP/IP package setup is complete at this point. Before we can fully set up NFS and YP, we need to build a new kernel. We'll go on to Step 15 and build a kernel, then use **sysadm setuppackage** again on NFS and YP. Then we'll reboot the system and complete the setup for YP.

Step 15: Building a Custom Kernel

We will build the kernel for the primary release in this step. Our homogeneous OS client, **dg1**, will be booting a slightly different configuration of this kernel. Note that client managers can build their own kernels or server managers can build kernels for clients. For now, we're interested in getting our server installed; in Step 19 we'll show you how to build kernels for clients.

To build or rebuild a kernel, you must edit the prototype system file to reflect what you actually have in your system. We will use **sysadm newdgux** to edit the system file, and run the build programs. When the process completes successfully, you'll reboot the system and bring up your custom kernel.

In this procedure, the **newdgux** command will ask you for the name of the system file that contains *your* changes to the system's parameters and a list of all devices on the system. The default name is the prototype system configuration file, **aviion** for servers and stand-alone machines. If you're configuring for a diskless client, you should specify **diskless**. You will then be asked to name an editor to edit the system file. The default is **vi**. When you finish editing, **config**(1M) automatically runs on the system file to produce program code in a file named **conf.c**. Next, a build program is invoked. This program compiles **conf.c** and links the libraries in **/usr/src/uts/aviion/lb** to build your customized kernel image. The executable image is built in **/var/Build**. If **config** or the build program are unsuccessful, the editor is invoked again.

In the example below, we will edit the file so that it represents our example system. The **newdgux** command concatenates the available prototype files together into a single file. In this case, we have files for the DG/UX operating system, TCP/IP, and NFS. If you had other products, then the prototype files for those products would also be concatenated. Read the explanation parts of the system file as you edit it. You will find typical configurations for diskless workstations and server/stand-alone machines.

We will use the # notation to comment out those items we do not want configured into our system. Bold #'s below are the ones we added to the file. We begin by typing

```
# sysadm newdgux ↵

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

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

Editor? [vi] ↵
```

```
#          Copyright (C) Data General Corporation, 1987 - 1989.
#          All Rights Reserved.
#          Licensed Material -- Property of Data General Corporation.
#          This software is made available solely pursuant to the
#          terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1989 system.dgux.proto     92.6"


#-------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       DG/UX
# (Release):            4.20
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for both workstations and server systems
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#

##### Typical workstation configuration:
```

NOTE: For tape and disks, you can use a star in the unit field to incorporate all devices of that type. For instance, **sd(insc(),*)** and **st(insc(),*)**.

```
#    kbd()            # -- keyboard
#    grfx()           # -- graphics display
#    duart()                    # -- integrated Duart terminal line controller
#    lp()             # -- integrated line printer controller
#    sd(insc(),*)     # -- all SCSI disks on integrated SCSI adapter
#    st(insc(),*)     # -- all SCSI tapes on integrated SCSI adapter
#    inen()           # -- integrated Ethernet controller
#
#    ptc()            # -- pseudo-terminal controller device
#    pts()            # -- pseudo-terminal slave device
#    pmt()            # -- pseudo-magtape device
#    log()            # -- Streams logger pseudo-device
#    prf()            # -- profiler pseudo-device
```

NOTE: For tape and disks, you can use a star in the unit field to incorporate all devices of that type. For instance, **sd(cisc(),*)** and **st(cisc(),*)**.

```
##### Typical server system configuration:

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

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

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Protocols:
```

Text skipped for brevity.

```
#-----------------------------------------------------------------
# STREAMS Modules:
```

Text skipped for brevity.

```
#---------------------------------------------------------------
# Tuneable Configuration Parameters:
```

Text skipped for brevity.

```
#   Parameter Name                  Value
#   --------------                  -----
#
    TZ                                          300
    NPROC                           256
    MAXUP                           64
    NODE                            "sales"
    MACH                            "AViiON"
```

NOTE: The following 6 parameters show the settings you might have on an OS server system. For a diskless client, you would uncomment the commented parameters and comment out the first one, the first DUMP parameter.

```
    DUMP                            "st(insc(),4)"
#   DUMP                            "inen()"

#   PERCENTNFS                                  100
#   NETBOOTDEV                                  "inen()"
#   ROOTFSTYPE                                  NETWORK_ROOT
#   SWAPDEVTYPE                                 NETWORK_SWAP

#
#---------------------------------------------------------------
#           Copyright (C) Data General Corporation, 1987 - 1989 - 1989.
#           All Rights Reserved.
#           Licensed Material -- Property of Data General Corporation.
#           This software is made available solely pursuant to the
#           terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1989 system.nfs.proto      92.3"


#---------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       NFS
# (Release):            4.20
#
```

Text skipped for brevity.

```
#---------------------------------------------------------------
#           Copyright (C) Data General Corporation, 1985 - 1989.
#           All Rights Reserved.
#          Licensed Material -- Property of Data General Corporation.
#          This software is made available solely pursuant to the
#          terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K    tcpip   90.1"


#---------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       TCP/IP
# (Release):            4.20
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#---------------------------------------------------------------


#---------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Verify typical configurations for both workstations and
# server systems. You will need at least one LAN controller
# (inen or hken).
#
# It is also recommended that you include the loopback pseudo-device.


    loop()
    hken()

#---------------------------------------------------------------
```

Text skipped for brevity.

```
#---------------------------------------------------------------



#---------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# When using more than 1 hken boards, it is advised that the
# configuration parameter  PERCENTSTR, defined in the dgux master
# file, be changed. The suggested value is 15.
#
#   Parameter Name                Value
#   --------------                -----
#
#
#----------------------------------------------------------------
```

When you have finished editing the file, exit **vi**. Next, you will see the following:

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

But if **config** encounters errors, you will see this:

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

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

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

But if the build fails, you will see the following:

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

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

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

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

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

If the build succeeds, you can install your new, customized kernel:

```
Install the New Kernel? [no] y ↵
For a diskless client? [no] ↵
Kernel Path Name? [/dgux.aviion] ↵

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

Setting Up NFS and YP

Now that we've built TCP/IP into the kernel, we can set up NFS and partially set up YP. By default, hosts are set up as YP clients. After we run **sysadm setuppackage** and reboot the kernel, we'll have to run a few commands to complete our set up as a YP master or a server.

```
# sysadm setuppackage ↵

Release Name? [PRIMARY] ↵
Package Name? [all] nfs ↵
Processing setup scripts for package nfs.

Setting up the rc#.d directory links.
Cleaning links in /etc/rc#.d directories.
Making rc script links to source directory.

That completes the automated portion of the nfs configuration.

setuppackage is finished.

#

# sysadm setuppackage ↵

Release Name? [PRIMARY] ↵
Package Name? [all] yp ↵
Setup package yp in MY_HOST root? [yes] ↵

Processing setup scripts for package yp.

Enter the name of the YP domainname: top.domain ↵

---- This host will first run as a YP client.
---- Setting YP domain to:  top.domain.
Is the domainname correct? y ↵

-- To initiate YP services you will have to change to init level 3.
-- To complete the YP setup as a YP server or master, please
   refer to the ONC/NFS release notice for this release.

setuppackage is finished

#
```

Now that your kernel has been built, we need to finish the setup steps for the Yellow Pages. We will do the following:

1) Reboot the system and come up to init level 3

2) edit **/etc/nfs.params** and set **ypserv_START=**"**MASTER**" and set **yppasswdd_ARG=**"**/etc/passwd -m passwd**"

3) run **/usr/etc/yp/ypinit -m**

Before doing these steps, see the ONC/NFS Release Notice and the manual *Managing NFS and Its Facilities on the DG/UX™ System.*

Remember, you must reboot your system after configuring the kernel to have TCP/IP and NFS services before completing the YP setup to master or server.

Step 16: Setting Default Boot Characteristics

Now that your kernel is configured, there are two boot characteristics that you can set: a default boot path in the SCM and a default initial run level.

The SCM Boot Path

You can set a default path for booting the DG/UX operating system. To go to the SCM, type:

```
# halt -q ↵

SCM> f ↵

View or Change System Configuration

1 Change boot parameters
2 Change terminal parameters
3 View memory configuration
4 Change testing parameters

Enter Choice(s) -> 1 ↵

Change Boot Parameters

1 Change system boot path
2 Change diagnostics boot path
3 Return to previous screen

Enter Choice(s) -> 1 ↵

Boot Path = [ ]
Do you want to modify the boot path? y ↵
Enter new boot path: cied()root:/dgux ↵
Boot path = [cied()root:/dgux]
Do you want to modify the boot path? n ↵

SCM>
```

Now that our default boot path has been set, we can simply type the single letter **b** to boot the operating system. To override the built-in root logical disk that gets mounted at boot time, use the **−a** option with the SCM boot command. When you provide this option, the kernel prompts you for all boot information. To boot with the disk **alt_root** as the root disk, use a command line like this:

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

The Initial Run Level

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

```
    def:3:initdefault
```

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

```
    SCM> b cied()root:dgux −s ↵
```

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

Step 17: Starting System Administration

This section is where we begin doing some of the more traditional system administration tasks. You can add user accounts, add terminals, add local and remote printers, and start system accounting programs. By setting some of these things up now, your system will be ready when terminal users and diskless client workstation users log on.

Adding User Accounts

All devices in the example system are now configured. Now you may want to add one or more users to your system. On a server and workstation client, you should have a login account for yourself in addition to the **sysadm** administrator's login. Since we have set up the OS server **sales** as the YP master, then all users added from **sales** will be part of the YP database. For our sample case, we would add the following users: the manager of **sales**, the manager of **dg1**, and later, the manager of **sun1**. Go to Chapter 14, "User Account Management" for information on setting user defaults and adding user accounts.

Setting Up Terminals and Printers

The procedure for adding tty lines depends on the number of tty entries that you have in **/dev**. If you have less than 64, then use this procedure as described below. If you have more than 64, then you need to run **sysadm newdgux** and change the NPROC variable to suit your needs. NPROC determines the maximum number of processes that can be active at one time on your system. If, for example, you have 84 ttys, then you need to adjust the NPROC variable upward by 20 from its current default value. This will prevent the process table from overflowing when processes are started on the ttys. After running **newdgux**, reboot your system to initialize the new kernel, then run **sysadm installtty** which spawns a **getty** process on every available tty line (all tty entries in **/dev**). If you have **getty** processes running on unused lines, you can edit **/etc/inittab** and change the "respawn" field to "off" for those you don't want activated.

The following example is for 24 ttys. To begin, type:

```
    # sysadm installtty ↵

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

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

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

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

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

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

Adding Line Printers

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

OS clients will access the server's line printer. Clients need to get the name of the server's printer and then use the **sysadm addlp** command and set up the server's printer as a remote printer.

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

```
# sysadm addlp ⮐
```

We complete the dialogue for the example system as follows:

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

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

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

Printer name?  mainlp ⮐
```

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

Printer device file?  list ↵

The available devices are:

tty00 through tty23

Printer device file?  tty00 ↵

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

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

The scheduler has been restarted.
```

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

```
# sysadm defaultlp ↵
```

The system responds as follows:

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

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

The new default printer is mainlp.
```

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

Starting the Accounting System

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

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

Phase Four: Adding OS Releases and Clients

A release must be added to the system before you can attach a client to that release. We have already installed our primary release and we will attach diskless client **dg1** to it. We still need to add a secondary release for our foreign diskless client **sun1**.

The steps in this phase are

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

Step 18: Adding Secondary Releases

OS release software consists of one or more software packages that are loaded into the same directory tree. For the example system, we need to use **sysadm addrelease** to create the appropriate directories for our secondary release, **68k_sunos_4.0**. Note that we already created a logical disk of 150,000 blocks for this release. Next, we'll use **sysadm loadpackage** to load software into the **/usr** directory for that release.

Here we'll create a secondary release area for our diskless client, **sun1**.

```
# sysadm addrelease ↵
```

The system responds as follows.

```
New Release Name? 68k_sunos_4.0 ↵
Usr Directory?  [/srv/release/68k_sunos_4.0/usr]  ↵
Share Directory? [/srv/share] ↵
Client Root Parent Directory? [/srv/release/68k_sunos_4.0]  ↵
Client Swap Directory? [/srv/swap] ↵

Release 68k_sunos_4.0 has been added. You may now use loadpackage.
```

The last system response tells us that the appropriate **sysadm** and directory entries have been made. Next, we need to load software into this newly created release area. As we did earlier, use **sysadm loadpackage** to load the software release package for **sun1**. Before attempting to add clients for a foreign release, consult the manuals supplied with the foreign release.

Step 19: Building Kernels for Diskless Clients

On Foreign Systems

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

On AViiON Systems

Although OS servers and OS clients can run the same primary release, the client boots a slightly different kernel than the server. Servers boot the starter kernel supplied with the primary release. After the server is up and running, the server must build a kernel for diskless clients. Diskless client kernels reside in **/srv/release/PRIMARY/root/_Kernels/dgux.diskless**.

After a client is up and running, the client may choose to build future kernels for itself, or have the server manager do it.

Let's configure a kernel for diskless client **dg1**.

```
System Name? [aviion]  diskless ↵
```

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

```
Editor?  [vi]  ↵
```

Edit the system file as before. When you have finished editing the file, exit **vi**. Next, you will see the following:

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

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

```
Install the new kernel? [n] y ↵
For a diskless client? [n]  ↵
Release Area? [PRIMARY] ↵
Kernel Path Name?[/srv/release/PRIMARY/root/_Kernels/dgux.diskless]↵
Save the old kernel? [y] ↵
Link all primary clients to the new kernel? [y] ↵
```

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

Step 20: Setting OS Client Defaults

The **sysadm clientdefaults** function records defaults for the **addclient** function. With **clientdefaults**, you can create sets of defaults to be used for different groups of diskless clients. That is, you might have a set called **dgset** for your Data General client machines running the primary release, and you might have a set called **sunset** for all those clients that will be running the **68k_sunos_4.0** release.

Since we know we'll be running two different releases, let's create a default set for each. We'll begin by typing:

```
# sysadm clientdefaults ↵
```

The system responds as follows:

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

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

Do the same for the foreign client **sun1**.

Step 21: Adding OS Clients

Now that the primary release is running on our OS server, we'll add our diskless OS client, **dg1**, to the primary release, and we'll add **sun1** to a secondary release. To add diskless clients, you must first add entries for those clients to the **/etc/host** and the **/etc/ethers** files. Do this with **sysadm addhost** and **addether**. The sequence of **sysadm** commands for adding clients is:

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

Adding Clients to /etc/hosts

To add our first client, we type:

```
# sysadm addhost ↵

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

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

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

Do the same for **sun1**.

Adding Clients to /etc/ethers

To add our client, we type:

```
# sysadm addether ↵

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

Do the same for **sun1**.

Adding an Example Client

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

We're ready to attach our client to its release. We begin by typing:

```
# sysadm addclient ↵
```

The system responds as follows:

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

Server and Client /etc/fstab Files

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

The **fstab** files for our example system look like the following:

Server: **sales** --

```
/dev/dsk/root                                /        dgux
/dev/dsk/usr                              /usr        dgux
/dev/dsk/swap                             swap        swap
/dev/dsk/srv                              /srv        swap
/dev/dsk/srv_swap                    /srv/swap        dgux
/dev/dsk/usr_opt_X11              /usr/opt/X11        dgux
/dev/dsk/usr_opt_imagen        /usr/opt/imagen        dgux
/dev/dsk/srv_dgux420       /srv/release/PRIMARY        dgux
/dev/dsk/srv_sunos4  /srv/release/68k_sunos_4.0        dgux
/dev/dsk/var_tmp                         /var/tmp     dgux
/dev/dsk/sales_accounts            /sales/accounts     dgux
```

Client: **dg1** --

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

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

Setting Up Packages on the Client

Back in Step 14 we set up TCP/IP, NFS, and YP for the server machine by running the **sysadm setuppackage** command. Similarly, a new client's packages must be set up. These must be set up on the client machine when it is first booted.

On the OS client machine, type the following:

```
# sysadm settuppackage ↵
```

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

Step 22: Booting and Setting Up an OS Client

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

Booting an Example Diskless Client

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

```
SCM> b inen() ↵

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

DG/UX Bootstrap Release 4.20 Version (diskless)

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

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

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

Setting Up Diskless Clients with sysadm

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

- Set their **inittab** default boot run level; it is shipped at level s, single user, but you may prefer to have your system come up in level 3, multiuser. We did this earlier for the server at the end of Step 15.
- As prompted when the system comes up, use **sysadm setuppackage** on the client machine. Note that by default, OS clients use the same **tcpip.params** setting as the OS server.
- Check **/etc/fstab** to see that all needed file systems are listed. Add as necessary and mount them with **sysadm fsmgmt**.
- Add a remote printer with **sysadm addlp**.

The End

This concludes the installation information on the DG/UX operating system. The rest of this manual is divided up per administration tasks. You'll find the information you need to fulfill your responsibilities as a system administrator. Bon voyage.

End of Chapter

Chapter 3
Operating the DG/UX System

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

The major sections of this chapter are:

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

Operation Terms

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

init	The **init(1M)** program creates all other processes based on instructions in the file **/etc/inittab**. **Init** is invoked in two ways: inside the DG/UX system as the last step in the boot procedure, and from the command line with a run level as argument. When invoked during the boot procedure, its first function is normally to **fork** a single superuser shell.
rc scripts	Run command scripts are executed at every boot. They kill or start system services as directed by **K** and S switches and ID numbers. 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 are active in any run level, those that are started with the **S** switch.

run level Also known as run state or run mode. Run levels are categories used to control *which* and *how many* system services are available. Those processes are invoked by **rc** scripts. A run level is indicated by the numbers 0 though 6, and S. For example, run level S is the single-user mode. In run level S, only the root file system is mounted and only a single superuser shell is running. See Table 3-1 for other run level descriptions.

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

SCM> The System Control Monitor prompt is displayed when you're communicating directly with the processor. From the SCM, you'll load the DG/UX system into main processor memory. You'll can also use the SCM to boot the system, or any other bootable files such as stand-alone **diskman** or diagnostics. See *Using the AViiON™ System Control Monitor (SCM)* for complete information.

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

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

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

DG/UX System Run Levels

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

Table 3-1 DG/UX Run levels

Run Level	Description
0	Currently undefined. Reserved for future use.
S	Single-user is a default, low-level run mode. This is the first mode the system enters upon booting. All processes spawned by **init** are killed. All file systems except **/** (root) and **usr** are unmounted.
1	Administrative mode is used to install and remove software utilities; run file system backups and restores; and to check file systems. Run level **1** unmounts everything except **/** (root) and **/usr** and kills all user processes, except those relating to the system console.
2	By default, **/** (root), **/usr**, and locally configured file systems are mounted in this run level. NFS services are not available in this level.
3	Multiuser/remote file-sharing mode. Same as level 3, but with full TCP/IP and NFS services.
4	User-defined run level. Data General does not supply **inittab** definitions for this state.
5	Currently undefined. Reserved for future use.
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 Table 3-2 for a complete list of the processes that start when you go to multiuser states 2 or 3.

When you bring up the DG/UX system in multiuser state:

- The **/** and **/usr** file systems are mounted.
- Local file systems are mounted in run level 2. Remote file systems are mounted in run level 3.
- The error daemon, **cron** daemon, and the **update** daemon are started.
- The LP system and UUCP are ready to use. The **getty** processes are spawned on all connected terminal lines in **/etc/inittab** that specify *respawn*. On timeshare systems, this means that users can now log in.
- If used, TCP/IP transmissions work outward in run level 2, and work in both directions in run level 3.

Operation Procedures

The following procedures are covered in this section:

- Start Up the System
- Shut Down the System
- Recover From System Failure
- Repair Damaged Root File System

Procedure 3.1: Start Up the System

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

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

```
SCM> reset ↵
SCM> b cied() ↵

root:cied(0,0)/dgux loading...


INIT: SINGLE USER MODE
```

For future operations, note that you can use the **boot** command to load any executable file you choose. For the above example, we loaded our standard operating system. If, for instance, you wanted to load a diagnostics program, you'd enter a command line like

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

The only other time that diagnostics are run is when you power up from a cold start. At that time, diagnostics run automatically followed by a root file system check done automatically by **fsck**.

Changing Run Levels

You can switch run levels *upward* with the **init(1M)** command. Whenever you are switching run levels *downward*, use the **shutdown(1M)** command. Note that you can shut down only to run level S or 1; use **init** to go back up to the desired run level as follows. Type the **init** command with a run level argument:

# **init 1**	Takes you to administrative mode.

# **init 2**	Takes you to multiuser mode.
# **init 3**	Takes you to multiuser mode with NFS.

Rebooting

To reboot the system, type

```
# halt -r ↵
```

to cause the system to shut down to the SCM and then automatically come back up to the default run level that is set in **/etc/inittab** (usually init 3). This is useful when you want to reboot the system, but don't want to monitor its output during the process. You could type the **reboot** command and come back later to a fully booted system.

When your system goes down, the autoboot process will automatically bring your system back up to the default run level set in **/etc/inittab**.

Procedure 3.2: Shut Down the System

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

There is no **sysadm** command to shut down the system. Do this from the command line with **shutdown(1M)**.

When we say shut down the system, we generally mean to go to a lower run level in order to perform some administrative task. Often, you will want to shut down to run level 1, administrative state. Other times, you may want to go down to run level S, single-user, to turn off all processer power.

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

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

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

Shut Down to Administrative Mode: Run Level 1

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

Options you can use are:

−y	Pre-answers the confirmation question so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.
−i1	Go to run level 1, administrative mode.
−g0	Allow a grace period of 0 seconds.

We type the following to shut down:

```
# shutdown −y −i1 −g0 ↵

Shutdown started.

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

INIT: New Run Level: 1
#
```

Now you are in run level 1, administrative mode. The **/** (root) and **/usr** 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.

Shut 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 and to the SCM. From the SCM, you can turn off power to the processer. Type:

```
# shutdown -g0 ↵

Wait for 'INIT: SINGLE USER MODE' before halting.

INIT: New run level: S

#
```

Ignore any prompts before the system arrives at single-user mode; they represent the current shell which is about to be killed. When the prompt appears *after* the `INIT: SINGLE USER MODE` message, you are in single-user mode. All other processes have been stopped. You can change run levels *upward* at this point with the **init(1M)** command, or, you can bring the system all the way down. Below, we'll take the system all the way down:

```
# halt -q ↵

CPU HALTED


SCM>
```

Now the CPU has been halted and you should have the SCM prompt. You can turn off the power now.

Procedure 3.3: Recover from System Failure

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

You're here because your system has suddenly stopped operating. We address three kinds of system trouble: a panic, a hang, or a power failure.

Panics and Hangs

A panic or a hang means the system has halted by itself. The reasons are often unknown; they can be caused by the hardware or the software. If your system is experiencing a panic, a message like the following is displayed:

```
DG/UX SYSTEM PANIC  PANIC CODE: 20001
```

Write down the exact message and panic number. If your system is experiencing a hang, you will not get a message. Usually, you'll notice that the system does not respond to user input, the screens are all frozen, etc.

Perform a System Dump

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

A system dump involves two dumps: system memory and system image. Both are required for Data General to adequately analyze the error conditions.

System Memory Dump

The way you begin dumping depends on what has happened to the system. If your system has panicked, the dump procedure will begin automatically. If your system is hung, you need to initiate the dump procedure by pressing the reset button. This will give you the SCM> prompt.

When you get the SCM> prompt, type the **START 1000** command to begin.

```
        SCM>  START 1000 ↵

        DG/UX System aborted by operator.

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

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

```
        Dump destination device? [0]  ↵

        Mount tape.  Type New Line when tape is ready. ↵

        Tape volume 1 completed.

        System Dump completed successfully.
```

Running fsck on the Root File System

As always, the **fsck** program will run automatically when the system is rebooted.

```
    SCM> reset  ↵
    SCM> boot cied() ↵



    INIT: SINGLE USER MODE
    #
```

Now you're back in single-user mode and ready to do the second part of the dump, the system image.

System Image Dump

In all cases, when you do a memory dump, you should also dump the tailored system image (usually named **/dgux**) that was running at the time of the system failure. This image contains vital information necessary to interpret the memory dump. The memory dump is useless without the system image. If there is room, dump the system image on the same tape as you dumped the memory. If not, two tapes are fine. Use the following format on both reel and cartridge tapes:

Tape file 0:	memory contents in memory dump format
Tape file 1:	system image in cpio format
Tape file 2:	other files, programs, etc., in cpio format

Do not use absolute pathnames, i.e., starting with /.

File 0

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

File 1

Use the command lines:

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

The tape will be rewound and positioned for file 2 to be written.

File 2

Use the command line:

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

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

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

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

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

Power Failure

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

Procedure 3.4: Repair Damaged Root File System

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

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

Since you cannot boot the damaged root, you need to run stand-alone **fsck** on it. If **fsck** cannot fix it, then you will have to load a new root file system onto the **swap** logical disk, and mount the damaged root under the temporary root file system. You can repair the damaged root if you first mount it on the new root. The following sections show you how to do this. You will go through the following steps:

1) Invoke stand-alone **diskman** from release media.

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

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

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

5) Mount **/usr**.

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

7) Reboot the system as usual.

Repairing the Root

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

```
5. Load the Root File System
```

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

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

Enter the Logical Disk Name: [root] swap ↵

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

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

Press New Line when ready to continue... ↵

Loading ...
```

When loading is complete, the system displays:

```
The Root File System has been loaded.

Press New Line when ready to continue.
```

You're ready to exit **diskman**. Type **q**.

Booting the Starter Kernel

Now, we'll reboot:

```
SCM> reset  ↵
SCM> boot swap:cied() ↵

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

INIT: SINGLE USER MODE
#
```

Now check the **/usr** file system:

```
# /sbin/fsck -xp   /dev/dsk/usr
```

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

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

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

Damaged Files

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

- **/dam_root/dgux**
- **/dam_root/etc/passwd**
- **/dam_root/etc/group**
- **/dam_root/etc/init**
- **/dam_root/etc/inittab**
- **/dam_root/etc/ioctl.syscon**
- **/dam_root/bin/sh**
- **/dam_root/bin/su**
- **/dam_root/etc/init.d** (Replace all entries in this directory.)

You have two methods for restoring these files:

1) Use "Procedure 7.10: Extract a Few Files from fsdump Tapes" to restore individual files (**sysadm filerestore**).

2) Use "Procedure 7.9: Restore File Systems from fsdump Tapes" to do a full restore (**sysadm frestore**).

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

Rebooting with a Repaired Root

After you've repaired or replaced the key files above, shutdown and reboot your system.

If Your Repaired Root Will Not Boot

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

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

2) Go to the File System Management Menu and select number 1, "Make a File System." Specify **root** as the logical disk. CAUTION: this will erase all data on the logical disk.

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

4) Load a new **/** (root) file system on the **root** logical disk. When loading is complete, use **sysadm filerestore** to restore the files listed earlier in "Damaged Files."

5) Run **shutdown -g0**.

6) Reboot the system.

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

Procedure 3.5: Log System Errors

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

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

```
selector    action
```

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

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

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

user	Messages generated by user processes.
kern	Messages generated by the kernel.
mail	The mail system.
daemon	System daemons such as **routed** and **ftpd**.
auth	The authorization system: **login, su,** and **getty**.
lpr	The printer spooling system.
cron	The **cron** or **at** facility.
local0-7	Reserved for local use.
mark	For timestamp messages produced internally by **syslogd**.
*	An asterisk indicates all facilities except the mark facility.

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

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

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

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

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

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

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

How Run Levels Are Set

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

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

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

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

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

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

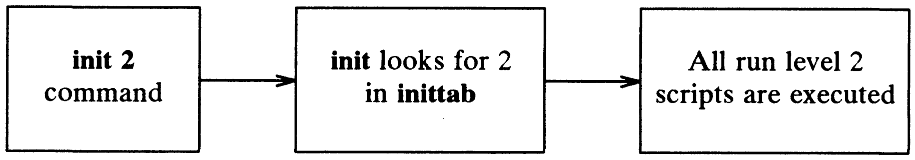

init 2 command
init looks for 2 in inittab
All run level 2 scripts are executed

Figure 3-1 An Example Run Level 2 Sequence

Run Command Scripts Per Run Level

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

The following table shows which scripts are started per run level. Note the cumulative effect: the higher the run level, the more processes running. Blanks indicate that a script is not running. From the table, we see that zero scripts are executed in run level S, 0, 5, and 6. Three scripts are executed in run level 1, eleven scripts in run level 2, and so on.

Table 3-2 RC Scripts Per Run Level

S	0	1	2	3	4	5	6
		rc.usrfs	rc.usrfs	rc.usrfs	rc.usrfs		
		rc.iacs	rc.iacs	rc.iacs	rc.iacs		
		rc.update	rc.update	rc.update	rc.update		
		rc.localfs	rc.localfs	rc.localfs	localfs		
		rc.setup	rc.setup	rc.setup	rc.setup		
			rc.userproc	rc.userproc	rc.userproc		
			rc.syslogd	rc.syslogd	rc.syslogd		
			rc.tcpipport	rc.tcpipport	rc.tcpipport		
			rc.ypserv	rc.ypserv	rc.ypserv		
			rc.account	rc.account	rc.account		
			rc.cron	rc.cron	rc.cron		
			rc.lpsched	rc.lpsched	rc.lpsched		
			rc.preserve	rc.preserve	rc.preserve		
			rc.x11	rc.x11	rc.x11		
				rc.tcpipserv	rc.tcpipserv		
				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" later in this chapter for a complete grid layout showing the kill/start mechanism for all scripts active at all run levels.

RC Scripts in /usr/sbin/init.d

rc.usrfs	Mounts (S) the **/usr** and **/tmp** (if you have one) file system in run levels 1, 2, 3, and 4. It unmounts (K) the same in run levels S, 0, 5, and 6. The **sysadm** program is available when **/usr** is mounted.
rc.iacs	Loads the IAC driver code once for run levels 1, 2, 3, and 4.
rc.userproc	Kills all user processes in run levels S, 0, 1, 5, and 6.
rc.setup	Displays packages that have not been set up at initial boot.
rc.sna	Starts up the Systems Network Architecture programs in levels 3 and 4.
rc.update	Starts the update daemon in run levels 1, 2, 3, and 4. The update daemon updates disks and preserves editor files left from the last boot.
rc.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.syslogd	Starts the **syslog** error logging program in run levels 2, 3, and 4; kills it in all other run levels.
rc.localfs	Mounts local (DG/UX) file systems listed in **/etc/fstab** in run levels 2, 3, and 4; unmounts them 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. **Lpsched** schedules requests taken by **lp(1)** for printing on line printers.
rc.preserve	Invokes the **expreserve(1M)** command in run levels 2, 3, and 4 to recover editor files saved during a system crash.
rc.ypserv	Starts the **yp** and **portmap** daemons, and sets the domain name in run levels 2 and 3; kills these in all other run levels.
rc.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	Starts the **telnetd, ftpd, tfpd, smtpd, rlogind, rwhod, rshd,** and **rexecd** daemons in run levels 3 and 4; kills them in all other run levels.

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

Check Scripts

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

The check scripts are:

chk.date	Displays the current system date and allows the administrator to set the correct date. A correct date setting is vital to ensure file creation and modification dates are correct. Also sets time zone based on **/etc/TIMEZONE** file.
chk.fsck	Runs **fsck** on all file systems listed in **/etc/fstab**. The **fsck** program is called with the **-xp** switch which checks file systems in parallel, and only checks those file systems that need checking.
chk.system	Performs the following system cleanup and initialization routines: • Reinitializes the **/etc/ps_data** file. • Cleans outs the **/var/spool/locks** used by the **uucp** program. • Makes a **/tmp** directory if one doesn't exist. • Runs the DG/UX setup scripts via the **init** command the first time the system is booted. • Checks for accounts without passwords.
chk.devlink	At the first **init** level change, this script automatically creates shortened names for devices in the sequence it finds them. For example, the first tape device will be device 0, the second will be device 1. These could then be specified as **/dev/rmt/0** and **/dev/rmt/1**. Device short names are taken from the **/etc/devlink** file.

Expert Run Level Information

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

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

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

The Fundamentals

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

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

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

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

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

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

The Sequence

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

1) You 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 **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

Since **init** relies on the information in **inittab**, let's look at a sample **/etc/inittab** file. The format of each line is:

id:level:action:process

- *id* is one or two characters that uniquely identify an entry.
- *level* is zero or more numbers and letters (s or 1 through 6) that determines what *level*(s) *action* is to take place in. If *level* is empty, the *action* is valid in all levels.
- *action* can be one of the following:

bootwait	Process the entry only at boot time. **Init** starts the process, waits for its termination and, when it dies, does not restart the process.
wait	When going to *level*, start *process* and wait until it's finished.
initdefault	When **init** starts, it will enter *level*. The *process* field for this *action* is not used.

once	Run *process* once and don't start it again if it finishes.
respawn	If *process* does not exist, start it, wait for it to finish, and then start another.
ondemand	Synonymous with **respawn**, but used only when **a**, **b,** or **c** is specified in *level.*
off	When in *level,* kill process or ignore it.

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

You can add a comment to the end of a line by prefacing the comment with #. Everything after a # on a line is ignored by the **init** program.

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

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

Figure 3-2 A Sample Inittab File

The first line in the sample file is:

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

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

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

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

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

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

RC Scripts and Check Scripts

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

chk.*	These scripts are usually run once, at boot time. An example is **chk.fsck** which runs the **fsck** program on file systems.
rc.*	These scripts are invoked with either a start or stop argument. An example is **rc.iacs** which starts or stops the Intelligent Asychronous Controllers.

Init.d Links

The above scripts exist in **/etc/init.d**, but they are invoked via links in an **/etc/rc***N***.d** directory. Remember, there are eight **/etc/rc***N***.d** directories: **/etc/rcS.d, /etc/rc0.d, /etc/rc1.d, /etc/rc2.d, /etc/rc3.3, /etc/rc4.d, /etc/rc5.d,** and **/etc/rc6.d**. The *names* of the links are labeled as follows:

S*000.name*

or

K*000.name*

The entries have three parts:

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

000	The next three characters are a number from 00 to 999. They indicate the order in which the files will be started (S111, S112, S113, etc.) or stopped (K231, K232, K233, etc).
name	The rest of entry is the script name in **/usr/sbin/init.d**.

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

For example, the run level 3 link name for **rc.localfs** is **S212.localfs**. This link is in **/etc/rc3.d**.

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

```
# cd /etc/rc3.d ↵
# ls ↵

K232.tcpipport  S111.usrfs      S233.ypserv     S352.nfsserv
S212.localfs    S232.tcpipport  S334.tcpipserv  S211.syslogd
S253.lpsched    S254.preserve   S113.update     S252.cron
S690.usrproc    S112.iacs       S251.account    S353.nfsfs
```

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

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

   rc.usrfs         111 K  K  S  S  S  S  K  K
   rc.iacs          112 K  K  S  S  S  S  K  K
   rc.update        113 K  K  S  S  S  S  K  K
   rc.setup         119 K  K  S  S  S  S  K  K

   rc.syslogd       211 K  K  K  S  S  S  K  K
   rc.localfs       212 K  K  K  S  S  S  K  K
#  rc.tcpipport     232 K  K  K  S  S  S  K  K
#  rc.ypserv        233 K  K  K  S  S  S  K  K
   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  K  S  S  S  K  K

#  rc.tcpipserv     334 K  K  K  K  S  S  K  K
#  rc.nfsserv       352 K  K  K  K  S  S  K  K
#  rc.nfsfs          353 K  K  K  K  S  S  K  K

#  rc.sna            371 K  K  K  K  S  S  K  K
#  rc.sna            372 K  K  K  K  S  S  K  K
#  rc.sna            373 K  K  K  K  S  S  K  K
#  rc.x11            391 K  K  K  K  S  S  K  K
   rc.userproc       690 K  K  K  S  S  S  K  K

# TCP/IP and NFS links are provided by those products'
# packages at installation.
```

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

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

Changing the Behavior of RC Scripts

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

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

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

```
biod_ARG="4"
```

Before changing any of these parameters, see the appropriate man page that explains the function of each parameter.

Adding Your Own RC Scripts

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

- Place the script in **/usr/sbin/init.d**.
- Link the script to files in appropriate run state directories using the naming convention described above. You can link each one manually or you can write a script to make all the links for you. (For instance **/usr/sbin/rc2.d/S222.***name* should be linked to **/usr/sbin/init.d/rc.***name*.)

- To differentiate your scripts, use three-digit numbers with the *middle* number always being even. Data General uses only odd middle numbers. Attach **S** (start) and **K** (kill) to your links.

- Create a copy of **/etc/dgux.rclinktab.proto** called (for example) **rc.linktab**; this serves as a record of *additions* to the **rc** scripts. Put your new scripts in this file, indicating the K and S switches in each run level.

Example: Adding Two RC Scripts

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

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

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

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

Figure 3-4 Your RC Script File

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

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

End of Chapter

Chapter 4 System Configuration Management

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

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

The major sections of this chapter are

- File Ownership and Access
- System Configuration Management Procedures
- General Operating Policy
- General Approach to Performance Management
- Tuning System Parameters

File Ownership and Access

In the DG/UX system, the access to any file is controlled by that file's owner. The owner with the widest range of ownership and privileges is **root**; it can override any permission settings and execute, open, read, delete, or change any file in the system. You can become **root** by executing the **su**(1) command with no arguments, or by logging in as root. Data General has created an administrative login that has all the privileges of **root**; this is the **sysadm** login. We recommend that you use the **sysadm** login instead of the **root** login. When you log in as **sysadm**, you're in the **/admin** home directory instead of in **/**. Using this login keeps your personal administrative files out of **/**.

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

Only **root, adm, uucp**, and **nuucp** are used to actually login. As **root** you can become **adm** or **nuucp**. The **uucp** login is used only by machines, that is, one machine logs on as **uucp** on another machine to set up a file transfer connection. See Chapter 12 for more on **uucp** and **nuucp**. The following table shows the administrative and system logins.

Table 4-1 Administrative and System Logins

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

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

System Configuration Management Procedures

The following procedures are covered in this chapter:

- Set system time and date
- Set tape archive defaults
- Recover forgotten root password
- Reconfigure the system

If you select the **sysmgmt** command from the **sysadm** Main Menu, the following choices are displayed:

```
                    System Configuration Management

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

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

Procedure 4.1 Set Time and Date

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

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

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

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

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

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

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


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

Procedure 4.2: Set Tape Defaults

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

Tapedefaults sets the defaults for using magnetic tape. The default information that you supply here is used by other processes such as when you modify dump cycles, dump files to tape, and restore files from backup tape. You need to supply the default tape drive and tape medium.

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

```
Default tape drive? 0 ↵

Default Tape Medium? qic 150↵

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

Procedure 4.3: Recover Forgotten Root Password

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

If you should forget the **root** password, this procedure shows you how to set a new root password.

There are two situations in which you may need to change the root password:

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

2) You are not currently logged on as root. For instance, you may be logged on as yourself and have the $ prompt. Because there is no way to access the root login, you'll have to bring the system down the hard way, sometimes called an "unclean" halt.

For the second possibility, go to the system console and do the following:

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

2) Type: **sync**

3) Wait five seconds, then press the reset button. This takes you to the SCM> prompt.

4) Run **fsck** on your root file system. Do this from your stand-alone **diskman** program. Run "Check a File System" from the File System Management Menu. This command runs the **fsck(1M)** program. If your stand-alone **diskman** is not on disk, you will have to load it from tape.

5) After you have checked and repaired your root file system, reboot your system.

6) In single-user mode, reset the root password with the **passwd(1)** command.

Procedure 4.4: Reconfigure the System

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

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

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

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

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

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

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

The **newdgux** command configures kernels for stand-alone, server, or diskless hosts. Stand-alone and server configuration is the same, while diskless configuration is slightly different. Diskless configurations use a different set of parameters. Some server administrators will configure kernels for diskless clients on the server host. Some diskless clients will configure their own kernels on the diskless host.

In this chapter, we build a stand-alone/server kernel, then we'll build a kernel for a diskless host. In the following example, we'll build a kernel for our example server. The **newdgux** command concatenates the available prototype files together into a single file. In this case, we have files for the DG/UX operating system, for TCP/IP, and for NFS. If you had other products, then the prototype files for those products would also be concatenated. Read the explanation parts of the system file as you edit it. You will find typical configurations for workstations and server/stand-alone machines.

We will use the # notation to comment out those items we do not want configured into our system. Bold #'s below indicate our edit. We begin by typing:

```
    # sysadm newdgux ↵

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

    Editor? [vi] ↵
```

```
#         Copyright (C) Data General Corporation, 1985 - 1989.
#         All Rights Reserved.
#         Licensed Material -- Property of Data General Corporation.
#         This software is made available solely pursuant to the
#         terms of a DGC license agreement which governs its use.

# sccsid = "@(#)   system.dgux.proto88.10"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       DG/UX
# (Release):            4.10
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-----------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for both workstations and server systems
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
#

##### Typical workstation configuration:
```

NOTE: For tape and disks, you can use a star in the unit field to incorporate all devices of that type. For instance, sd(insc(),*) and st(insc(),*).

```
#    kbd()          #-- keyboard
#    grfx()         #-- graphics display
#    duart()              #-- integrated Duart terminal line controller
#    sd(insc(),0)         #-- SCSI disk on integrated SCSI adapter
#    st(insc(),4)         #-- SCSI tape on integrated SCSI adapter
#
#    ptc()          #-- pseudo-terminal controller device
#    pts()          #-- pseudo-terminal slave device
#    pmt()          #-- pseudo-magtape device
#    plm()          #-- network lock manager pseudo-device
#    log()          #-- Streams logger pseudo-device
#    prf()          #-- profiler pseudo-device


##### Typical server system configuration:

    duart()              #-- integrated Duart terminal line controller
    cisc()               #-- SCSI adapter (on VME bus)
    cird()               #-- Ciprico Rimfire or SMD disk controller
    st(cisc(),4)         #-- SCSI tape on integrated SCSI adapter
    syac()               #-- Systech terminal line controller
    hken()               #-- Hawk Ethernet controller

    ptc()                #-- pseudo-terminal controller device
    pts()                #-- pseudo-terminal slave device
    pmt()                #-- pseudo-magtape device
    plm()                #-- network lock manager pseudo-device
    log()                #-- Streams logger pseudo-device
    prf()                #-- profiler pseudo-device

#
```

```
#-----------------------------------------------------------------

#-----------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# You should set the TZ variable to accurately reflect your timezone
# (300 minutes west of GMT is USA Eastern time).
#
# You should set the NPROC variable to the maximum number of processes
# that your system will be able to have.
#
# You should set the NODE variable to control your nodename for uname(1)
# and uucp(1), but not more than 255 characters.
#
# You should set the MACH variable to record the model of machine you
# are using, but not more than 255 characters.
#
# You should set the DUMP variable to the name of the tape device (in
# DG/UX Common Device Specification Format) that will be the default
# device to take dumps in case of system emergencies.
#
#------------------------------------------------------------------
# If your system is a diskless workstation, you should set the
# PERCENTNFS variable to 100 in order to get the best possible NFS
# performance.
#
# If your system is a diskless workstation, you must set the INIT
# variable to &init_diskless_run_etc_init.
#
#
#   Parameter Name                  Value
#   --------------                  -----
    TZ                          300
    NPROC                           256
    NODE                            "sales"
    MACH                            "AViiON"
    DUMP                            "st@(insc(),4)"
# PERCENTNFS              100
# INIT                              &init_diskless_run_etc_init

#
#-----------------------------------------------------------------
```

```
#          Copyright (C) Data General Corporation, 1985 - 1989.
#          All Rights Reserved.
#          Licensed Material -- Property of Data General Corporation.
#          This software is made available solely pursuant to the
#          terms of a DGC license agreement which governs its use.
#----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       NFS
# (Release):            4.10
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#----------------------------------------------------------------

#----------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# To use NFS, you must specify the NFS variable so that its implied
# value will be used.
#
#   Parameter Name                Value
#   --------------                -----
    NFS
#
#----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       TCP/IP
# (Release):            4.10
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for both workstations and server systems
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
# You will need at least one LAN controller (inen or hken).
# It is recommended that you include the loopback pseudo-device.

##### Typical workstation configuration:

#           loop()
#           inen()                  #-- integrated ethernet controller

##### Typical server system configuration:

            loop()
            hken()                  #-- Hawk ethernet controller
#
#-----------------------------------------------------------------

#-----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You will need the tcp, ip, udp and icmp protocols.
#
#   Protocol Name
#   -------------
    icmp
    udp
    tcp
    ip
#
#-----------------------------------------------------------------

#-----------------------------------------------------------------
# STREAMS Modules:
```

```
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
#
#
#   STREAMS Module Name
#   -------------------
            tcpip
            arp
            ether
            netlog
#
#----------------------------------------------------------------
#           Copyright (C) Data General Corporation, 1985 - 1989.
#           All Rights Reserved.
#           Licensed Material -- Property of Data General Corporation.
#           This software is made available solely pursuant to the
#           terms of a DGC license agreement which governs its use.

# sccsid = "@(#)   system.nfs.proto 88.2"
#----------------------------------------------------------------
```

When you have finished editing the file, exit **vi**. You will see the following:

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

But if **config** encounters errors, you will see this:

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

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

```
Sysadm will now attempt to build a kernel.
This may take a while.  Building ...
.
.
.
```

```
The build succeeded.
```

But if the build fails, you will see the following:

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

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

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

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

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

If the build succeeds, you can install your new, customized kernel:

```
Install the New Kernel? [no] y ↵
For a diskless client? [no]  ↵
Kernel Path Name? [/srv/release/primary/root/_Kernels/dgux.diskless]

Save the old kernel? [y]  ↵
Link /dgux to the New Kernel? [y]  ↵
```

The new kernel will not take effect until you shutdown the system to the SCM and reboot. Use the **shutdown**(1M) command, then reboot.

Building Client Kernels

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

```
System Name? [aviion]  diskless ↵
```

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

```
Editor?  [vi]  ↵
```

Edit the system file, but this time comment out the server items. Be sure to include the PERCENTNFS and INIT entries for diskless clients in the tuneable configuration parameter section. When you have finished editing the file, exit **vi**. Next, you will see the following:

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

When the build concludes, the new kernel is installed in a location accessible to the diskless client.

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

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

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

Configuration Error Messages

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

Category	Message
Cannot open a file or directory. Make sure the master file is in the proper directory and that it is named correctly.	Cannot open the master file [*master_filename*]. Cannot open master file directory.
The file in the master directory is not a legal master file.	No section definition found in master file [*master_filename*]. This file will be ignored.
There may be incorrect information in your system file, i.e., you may have misspelled a device name. Check that entries in your system file match those in your master file. Keyword errors pertain to the system file. Device flag and flag errors pertain to the master file.	Unknown Keyword: [*keyword_name*] Unknown Device Flag: [*device_flag*] Unknown Flag: [*flag*]
Cannot allocate space for internal structures. This is the result of an error returned from **malloc(3C)**. This is related to user logical address space. Check the master file directory for duplicate files.	Cannot Allocate Space. Allocate device entry: Out of memory. Allocate stream entry: Out of memory. Allocate protocol entry: Out of memory. Cannot allocate an alias structure. Cannot allocate a keyword structure. Error allocating Configured Device entry: Out of memory.
Illegal arguments sent to **config**.	Usage: config [-t] [-c] [-m master_file_directory] [-c conf_file] system_file

Illegal format for a master file or system file line. Device code errors and keyword errors are associated with the system file. The other errors in this category are associated with the master file.	Illegal Master file line: [*line*] Illegal protocol number: [*protocol number*] Illegal Domain number: [*domain number*] Illegal Socket number: [*socket number*] Illegal Device Code: [*device code*] No value associated with the keyword: [*keyword_name*]. Keyword will be ignored.
Two devices share the same major number. Change the incorrect one in the master file.	Warning: Device [*device_name*] and [*device_name*] have the same major number [*number*]. Warning: Device [*device_name*] on major number [*number*] configured.

General Operating Policy

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

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

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

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

Maintaining a System Log

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

- What devices are configured into the current kernel
- Equipment and system configuration changes (dates and actions)
- A record of system panics and hangs
- Maintenance records (dates and actions)
- A record of recurring problems and fixes.

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

General Approach to Performance Management

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

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

File System Organization

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

Maximizing System Usage

To ensure maximum system performance, you should check for

- Less important jobs interfering with more important jobs
- Unnecessary activities being carried out
- Scheduling of selected jobs for when the system is not so busy
- The efficiency of user-defined features, such as **.profile** and $PATH

Getting Process Information

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

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

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

Checking User $PATH Variables

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

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

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

- Path Efficiency

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

- Convenience and Human Factors

  Users may prefer to have the current directory listed first in the path (**:/bin**).

- Path Length

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

- Large Directory Searches

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

Shift Workload to Off-Peak Hours

Examine **/var/spool/crontab** to see if jobs are queued up for peak periods that might better be run at times when the system is idle. Use the **ps** command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands (such as **nroff**(1) or **troff**(1) at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(1) or **batch**(1) commands.

Tuning System Parameters

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

Uname Configuration Variables

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

Table 4-2 Uname Configuration Variables

Parameter	Default Value
NODE	"no_node"
MACH	"AViiON"
SYS	"dgux"
VER	"4.10"
REL	"00"

NODE	The UUCP node name of the system (sales, ABC, xyz31, etc.)
MACH	The name of the system's underlying hardware.
SYS	The name of the operating system.
VER	The version number of the operating system.
REL	The number of the system's release.

Setup and Initialization Configuration Variables

These variables are also listed in **/usr/etc/master.d/dgux**. They set the following system initialization parameters:

Table 4-3 Setup and Initialization Configuration Variables

Parameter	Default Value
DST	1
TZ	300
DUMP	"st(cisc(0),4)"
DEBUGGER	&sc_null_debugger
INIT	&init_run_etc_init
REGDISKS	1
STARTER	0

DST	Specifies the type of Daylight Savings Time being used. The different types are defined in **/usr/include/sys/time.h**.
TZ	Represents the timezone of your system in minutes west of Greenwich Mean Time. Set this according to your time zone. The default is USA Eastern time: 300 minutes west of GMT.
DUMP	A string holding the name of the default system dump device in DG/UX common device specification format. This device is the default tape device used to do a system memory dump after a PANIC or a halt. Set this to your primary tape device.
DEBUGGER	Represents the kernel debugger to be used by your system. The default is the null debugger stub. No other debuggers are currently available.
INIT	Represents the internal function that is executed upon system booting. The default function calls **/etc/init**. Do not change this parameter.
REGDISKS	A Boolean variable indicating whether or not the system will attempt to register all of its physical disks upon booting. The default is 1 (TRUE). Use 0 for FALSE. We recommend that you use the default.
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 recommed that you use the default.

CPU and Process Configuration Variables

These variables are also listed in **/usr/etc/master.d/dgux**. They set the following parameters:

Table 4-4 CPU and Process Configuration Variables

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

NPROC	Specifies the maximum number of processes the system can have at one time. For various sized systems use the following values: small 96; medium (default) 256; and large 512. The overall number of processes needed depends on the overall size of the system; specifically on the number of terminal lines available, the number of processes spawned by each user, and the number of system processes and network daemons. If the maximum number of processes is used up, the **fork(2)** system call will return the error EAGAIN.
NCPUS	Specifies the number of processors to run. If set to 0 (the default), all available CPUs will be used. Any other value specifies that number of CPUs to run. If the value specified is more or less than the number of CPUs present, a message to that effect is printed when the kernel is booted. Note that on a uniprocessor system, this parameter has no real effect since the one processor will always be run.
MAXSLICE	Specifies the maximum time in milliseconds a user process can run before being suspended. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE number of milliseconds. MAXSLICE is normally 1/2 second (500 milliseconds).
MAXUP	Specifies the maximum number of processes that a non-superuser can have in existence at one time. The entry is normally in the range of 15 to 25. This value should not exceed the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if ten people logged in on the same user ID, the default limit would be reached very quickly.

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

Pseudo-Device Unit Count Variable

These configuration variables set the number of units a specified pseudo-device will have. (No count variables are needed for real devices; any units present are useable.) Currently, there is only one variable in this category.

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

File System Configuration Variables

These variables are also listed in **/usr/etc/master.d/dgux**. They set the following file system parameters:

Table 4-5 File System Configuration Variables

Parameter	Default Value
ACCTON	5
ACCTOFF	2
PERCENTBUF	10
PERCENTNFS	75
CDLIMIT	2147483648
FREEINODE	4
FREERNODE	4

ACCTON Specifies the minimum percentage of free space required in the file system to which accounting records are currently directed. An accounting file is specified with the **acct(2)** system call and an accounting record is written to that file upon each process termination.

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

PERCENTBUF	Specifies a percentage of main memory (after initialization) to reserve for I/O buffers. The I/O buffers form a data cache containing disk file information. For various sized systems use the following values: small to medium (default) 10 and large 15.
PERCENTNFS	Specifies the percentage of system buffers that are available for NFS clients. This value will vary from the default (75) depending on your NFS needs.
CDLIMIT	Specifies the maximum size in bytes that a non-superuser file may attain.
FREEINODE	Specifies the maximum ratio of in-use inodes to free inodes in the system.
FREERNODE	Specifies the maximum ratio of in-use rnodes to free rnodes in the system.

STREAMS Configuration Variables

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

Table 4-6 STREAMS Configuration Variables

Parameter	Default Value
PERCENTSTR	8
STRLOFRAC	80
STRMEDFRAC	90
NQUEUE	100
NSTRPUSH	9
STRMSGSZ	4096
STRMCTLSZ	1024

PERCENTSTR	Specifies the percentage of system memory (after initialization) that is reserved for STREAMS buffers.
STRLOFRAC	Specifies the threshold percentage of in-use STREAMS buffers beyond which low-priority requests for STREAMS buffers will be denied. This variable is included to help prevent deadlock by starving out low-priority activity. The recommend value of 80 works well for current applications. This parameter must always be in the range 0 <= STRLOFRAC <= STRMEDFRAC.
STRMEDFRAC	Specifies the threshhold percentage of in-use STREAMS buffers beyond which medium-priority requests for STREAMS buffers will be denied. This parameter must always be in the range STRLOFRAC <= STRMEDFRAC <= 100.
NQUEUE	Specifies the maximum number of STREAMS queues (Streams plus instances of STREAMS modules) that may exist at any one time on the system. A minimal stream contains two queue pairs: one for the Stream head and one for the driver. Each instance of a module on a Stream requires an additional queue pair.
NSTRPUSH	Specifies the maximum number of STREAMS modules that may be pushed on any one Stream. This is used to prevent an errant user process from consuming all the available queue pairs on a single STREAMS module.
STRMSGSZ	Specifies the maximum number of bytes allowed in the data portion of a STREAMS message. A module maximum packet size of INFPSZ defaults the maximum packet size to this value.

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

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

Semaphore Configuration Variables

These parameter variables are also listed in **/etc/master.d/dgux**. The following variables are associated with interprocess communication (IPC) semaphores:

Table 4-7 Semaphore Configuration Variables

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

SEMMNI	Specifies the maximum number of unique semaphore sets that may be active at any one time on the system.
SEMMSL	Specifies the maximum number of semaphores that a semaphore set may contain.
SEMOPM	Specifies the maximum number of semaphore operations that can be executed per **semop(2)** system call.
SEMVMX	Specifies the maximum value a semaphore may have. The default is the maximum value for this parameter.
SEMUME	Specifies the maximum number of undo entries per undo structure.
SEMAEM	Specifies the adjustment on exit for maximum value. 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.

Shared Memory Configuration Variables

The following tunable parameters are associated with inter-process communication shared memory. These parameters are also defined in the **/usr/etc/master.d/kernel** file.

Table 4-8 Shared Memory Configuration Variables

Parameter	Default Value
SHMMNI	100
SHMSEG	6
SHMMAX	131072
SHMMIN	1

SHMMAX	Specifies the maximum shared memory segment size. The default value is 131072.
SHMMIN	Specifies the minimum shared memory segment size. The default value is 1.
SHMMNI	Specifies the maximum number of shared memory identifiers system wide. The default value is 100. Each entry contains 52 bytes.
SHMSEG	Specifies the number of attached shared memory segments per process. The default value is 6. The maximum value is 15.

Message Configuration Variables

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

Table 4-9 Message Configuration Variables

Parameter	Default Value
MSGMNI	50
MSGTQL	1024
MSGMNB	4096
MSGMAX	2048

MSGMNI Specifies the maximum number of message queues that may exist in the system at one time.

MSGTQL Specifies the maximum number of outstanding messages that may exist in the system at one time.

MSGMNB Specifies the maximum number of bytes that a message queue may contain.

MSGMAX Specifies the maximum number of bytes that a message may contain.

End of Chapter

Chapter 5 Release Management

This chapter is about managing releases. A release is the software that provides operating system service. The location into which a release is loaded makes up what we call the *release area.* Once a release area has been created, software can be loaded into it.

We distinguish between the *primary* release and *secondary* releases. The primary release (for instance the DG/UX 4.10 OS), runs as the main operating system on servers and stand-alone machines. Any other releases are secondary. You load the primary release with the initial loading tool, **diskman**; we showed this in Chapter 2. Secondary releases are loaded with **sysadm**. The primary release resides in **/usr**. The secondary release resides in **/srv**. The primary release is accessed via **/srv/release/PRIMARY** and the secondary release is accessed via **/srv/release/***release_name*. There is only one copy of **/usr** on the server (or on stand-alone machines). The **/usr** filesystem stores the host-independent programs and data files. Diskless clients have their own root file systems, a prototype of the root file system. The **sysadm addclient** function makes one copy (from **/usr/root.proto**) for each diskless client.

The remainder of this chapter contains **sysadm** procedures for managing releases. You can

- Create a software release area.
- Delete a release directory tree.
- Display information on existing release areas.
- Load software into a release area.
- Set up software in a release area.
- Create the **srv** directory tree.
- List tape table of contents.

Procedure 5.1: Create a New Software Release Area

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

A release is a collection of software packages intended for a specific architecture and operating system. To add a release means to create the appropriate directories and files that will be used by the release. Once a release has been added, you can load software into it.

When adding a new release, you'll use the following **sysadm** functions:

1) **makesrv** (You'll be prompted to run this command as needed.)
2) **addrelease**
3) **loadpackage**
4) **setuppackage**

Below, let's add a secondary release named 88k_dgux_5.0. We start by typing the following:

```
# sysadm addrelease ↵
```

The system responds as follows.

```
New Release Name? 88k_dgux_5.0
Usr Directory?  /srv/88k_dgux_5.0
Share Directory? [/srv/share] ↵
Client Root Parent Directory? [/srv/release/88k_dgux_5.0]  ↵
Client Swap Directory? [/srv/swap] ↵
Release 88k_dgux_5.0 has been added.
```

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

usr	/srv/release/*release_name*/usr

root proto	/srv/release/*release_name*/usr/root.proto
share	/srv/share
client root parent	/srv/release/*release_name*/root
client swap	/srv/release/*release_name*/swap

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

Procedure 5.2: Delete a Release Area

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

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

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

```
Release Name?  88k_foo_9.0 ↵
Do you really want to delete 88k_foo_9.0? [no] y ↵
Release 88k_foo_9.0 has been deleted.
```

Deleting this release removes the following:

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

Procedure 5.3: List Release Information

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

Listing releases consists of printing an entry for each release area with the release name, directory path names, installed packages, and attached clients. You can select information on all releases or on a specific release. Let's do both below. First, we'll display information on all releases. After that, we'll display information on a specific release, 88k_dgux_5.0.

```
Release Name? [all] ⤸

Release Name              Usr Path Name
------------              -------------
88k_dgux_5.0             /srv/release/88k_dgux_5.0
88k_foo_9.0              /srv/release/88k_foo_9.0
```

For the specific release:

```
Release Name? [all] 88k_dgux_5.0 ⤸

Release Name:           88k_dgux_5.0
Usr Directory:         /srv/release/88k_dgux_5.0
Root Directories:      /srv/release/88k_dgux_5.0
Swap Files:            /srv/swap

Packages:              V-windows 2.0
                       Zapfiles  1.0

Clients:               dg1
```

Procedure 5.4: Load Software into a Release Area

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

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

Software is loaded into **/**, **/usr**, or a subdirectory of **/usr** such as **/usr/opt**. When you add software to a release that is running on diskless clients, a copy goes into each diskless client root. This may include the server's root.

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

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

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

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

         X11_file1
         X11_file2

loadpackage for X11 is finished.
```

Procedure 5.5: Set Up Software in a Release Area

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

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

```
# sysadm setuppackage ↵
```

The system responds as follows:

```
Release Area? [PRIMARY] ↵

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

xray-vision   telepathy_1   fortune_teller

Package Name? [all] xray-vision ↵

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

At this point, you would begin responding to the queries specific to the given setup script. When you finish setting up a package, **setuppackage** looks to see if any other packages need to be set up. If not, the function exits.

Procedure 5.6: Create the srv Directory Tree

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

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

```
# sysadm makesrv ⮐

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

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

Procedure 5.7: List Tape Table of Contents

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

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

```
Tape Drive? [0] ↵
Is the tape mounted and ready? [yes] ↵

          <table of contents lists here>
```

End of Chapter

Chapter 6
Client Management

This chapter contains the **sysadm** procedures for managing diskless clients. You can add clients, delete clients, list clients, create a set of client defaults, and designate a default boot path for those clients that are attached to more than one release. See Chapter 1, Part 2 for a general discussion of servers, clients, netbooting, and the servnet.

Add diskless clients to a server in this sequence:

1) Use **sysadm clientdefaults**.
2) Use **sysadm addhosts** for each client.
3) Use **sysadm addether** for each client.
4) Use **sysadm addclient** for each client.
5) Set up client packages on the client machine.
6) Boot clients.

NOTE:

Because of basic operating system differences, the **sysadm** procedures for adding DG/UX OS clients may not completely set up foreign OS clients. AViiON servers will support foreign OS clients, but Data General can not supply the foreign system specific information necessary for a complete setup; this must come from a foreign system's documentation.

Procedure 6.1: Attach a Client to a Release

Purpose	This function associates a client workstation to a specific operating system that resides on a remote server's physical disk.
Starting Conditions	init 1 or higher You should have used **sysadm addhosts** and **addether** before using **addclient**.
sysadm menu	clientmgmt
Commands	**addclient**

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

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

Let's add an example client, a Data General diskless workstation named **dg2** that will be running the primary OS. We begin by typing the following:

```
# sysadm addclient ↵

Client Host Name? dg2 ↵
Defaults set name? [dgset] ↵
Use all defaults from dgset? yes ↵
Creating client root.
Creating client swap file.
Creating client /etc/fstab.
Creating client /etc/hosts.
Creating client /etc/tcpip.params.
Creating client /etc/nfs.params.
Client dg2 has been added. The client may now be booted.
Do you wish to add another client? [yes]  no ↵
```

Files Created by sysadm addclient

When you add a client, that client inherits the server's environment. This means that the client receives copies of the server's various parameter files: **dgux.params, tcpip.params, nfs.params,** etc. Clients can modify these as they want. When you add a client with **sysadm addclient**, the following directories and files are created.

Client Root

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

Client Swap File

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

Client Parameter and Data Files

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

fstab

The **addclient** function adds the following entries to the client **fstab:**

```
sales:/srv/release/PRIMARY/root/client_name
sales:/usr
sales:/srv/swap/client_name
sales:/sales/accounts (Our example)
```

This file must contain entries for all other file systems that a client needs to access. See **fstab**(1M).

hosts

The **addclient** function adds *client_name* to **/etc/hosts**.

tcpip.params

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

nfs.params

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

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

Client Kernel Link

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

Server Bootstrap Link

A diskless client uses a secondary bootstrap to load **dgux.diskless** over the network. This bootstrap file is **/usr/stand/boot.aviion**. The **addclient** function makes a symbolic link from **/usr/stand/boot.aviion** to **/tftpboot/***client_ethernet_addr*.

Server bootparams File

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

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

Server exports File

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

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

Client/Release Data Files for sysadm

The **addclient** function creates **/srv/admin/clients** and **/srv/admin/releases**. These directories contain files used by the **sysadm** program.

The **clients** directory contains data files for the server and all clients. Here is an example client file:

```
CLIENT_NAME=dg1
CLIENT_ROOT_SIZE=5m
CLIENT_SWAP_SIZE=8388608
CLIENT_HOME=/sales/accounts
CLIENT_BOOTSTRAP=/srv/release/PRIMARY/root/_Kernels/dgux.diskless
```

The **releases** directory contains data files for all releases on the system. Here is an example file for the primary release:

```
REL_NAME=PRIMARY
REL_USR=/usr
REL_SHARE=/srv/share
REL_ROOT=/srv/release/PRIMARY/root
REL_SWAP=/srv/swap
```

Procedure 6.2: Delete a Client From a Release

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

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

```
# sysadm delclient ↵
```

The system responds as follows:

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

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

Client dg2 has been deleted from PRIMARY.

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

Procedure 6.3: List Client Information

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

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

Begin by typing:

```
# sysadm lsclient ↵
```

The system responds as follows:

```
Client Host Name? all ↵

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

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

```
Client Name?  dg1 ↵

Client Name:        dg1
Release Name:       88k_dgux_4.10
Root Directory:     /srv/release/PRIMARY/root/dg1
Root Size:          40,000 blocks
Swap File:          /srv/swap/dg1
Swap Size:          32,000 blocks
Kernel File:        /srv/release/PRIMARY/root/_Kernels/dgux.diskless
```

Procedure 6.4: Set Client Defaults

Purpose	This procedure sets the defaults for the **addclient** function.
Starting Conditions	init 1 or higher
sysadm menu	clientmgmt
Note	The primary release is the one running on the OS server, for instance, 88k_dgux_4.10.
Commands	**clientdefaults**

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

This function records defaults that will be used by the **addclient** function when you are adding diskless clients to your system. This function displays the names of any existing sets and allows you to define others. Assume we already have two sets defined, *68kset* and *88kset*.

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

```
# sysadm clientdefaults ↵
```

The system responds as follows:

```
The current set names are:

68kset    88kset

Defaults Set Name? dgset ↵
Default Release Area? PRIMARY ↵

Default Swap Size? [16m]  ↵

Default Home Directory? [/home] /sales/accounts ↵
Default Kernel? [/srv/release/PRIMARY/_Kernels/dgux.diskless] ↵

Defaults for Set dgset have been assigned.
```

Procedure 6.5: Set a Client's Default Boot Path

Purpose	Identifies which release a client will boot by default.
Starting Conditions	init 1 or higher
sysadm menu	clientmgmt
Commands	**bootdefault**

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

```
# sysadm bootdefault ↵
```

The system responds as follows:

```
Client Host Name? dg2 ↵

Boot Release Area?88k_foo_9.0 ↵

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

End of Chapter

Chapter 7
Disk Management

Diskman is a program for managing your physical and logical disks. It is composed of multilayered menus which call up DG/UX programs to perform various disk tasks.

The major sections of this chapter are

- Invoking the Diskman Program
- Using Diskman Menus
- Disk Management Procedures
- Command Line Options

Invoking the Diskman Program

The DG/UX system comes with two versions of **diskman**:

- **Stand-alone**: Invoke this version directly from tape when you are installing the DG/UX system, or boot the disk file **/usr/stand/diskman** from your **root** logical disk.
- **Stand-among**: Invoke this version through the **sysadm diskmgmt** command or directly from the shell as superuser. See "Command Line Options" at the end of this chapter.

Chapter 2 contains valuable definitions and explanations of physical and logical disks. Chapter 7 details the association of file systems with logical disks. Be sure that you are thoroughly familiar with these concepts before you alter disks.

Basic Diskman Functions

Some of the major tasks you can do with **diskman** are

- Format physical disks.

- Create, delete, and copy logical disks.
- Display physical and logical disk information.
- Create, check, and repair file systems.

Using Diskman Menus

The **diskman** program works very much like **sysadm**; **diskman** menus lead you through the functions, query for information, respond to selections, and display error messages. At the end of a function, you are returned to the previous menu. When you are offered a default, simply press the New Line key to choose it. Table 7-1 shows how to use **diskman**. Remember: HELP is available by typing the question mark, ?. Use it freely.

Table 7-1 How to Use Diskman Menus

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

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

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

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

The **diskman** program is quite similar to **sysadm**. Let's go through each of the Main Menu choices. The following sections show all the options of **diskman**.

Disk Management Procedures

```
                    Diskman Main Menu

1. Physical Disk Management Menu

2. Logical Disk Management Menu

3. File System Management Menu

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

Enter Choice:
```

NOTE: This chapter does not give you the information you will need for initial installation. See Chapter 2 for instructions on using the stand-alone version of **diskman** to install the DG/UX system.

Overview: From Physical Disk to File System

Below, we use **comm** as an example to show a general overview of going from an unformatted physical disk to a useable file system. This involves using information from this chapter, then moving to Chapter 8 to complete the process.

1) Format the physical disk.
2) Register the physical disk.
3) Create a logical disk, named **comm**, on the physical disk.
4) Create a file system on the logical disk.
5) You are finished with **diskman**. GO TO Chapter 8.
6) Using **addfsys**, add the file system to **/etc/fstab** and create a mount directory named **/comm**.
7) Mount the file system on directory **/comm** and it is accessible to users.

Items 1 through 5 are explained in more detail in the procedure sections that follow in this chapter. Items 6 through 8 are explained in Procedure 8.1.

Procedure 7.1: Physical Disk Management

As always, if you are not sure how to respond to any query, type ?. Let's begin looking at the **diskman** menus, starting with selection 1:

```
              Physical Disk Management Menu

1. Register, Deregister, or List Registered Physical Disks
2. Add, Recover, or Display Bad Blocks on a Physical Disk
3. Display a Physical Disk's Layout
4. Display a Physical Disk's Label
5. Format a Physical Disk

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

Enter Choice:
```

Procedure 7.1.1: Register, Deregister, or List Registered Physical Disks

Purpose	To register a physical disk so that logical disks associated with that physical disk are available for use. To deregister a physical disk no longer in use. To list the physical disks that are already registered.
diskman menu	Physical Disk Management Menu
References	**diskman(1M)**

You should register any new disk you add and deregister any that you remove. A registration works like a link. When you register a physical disk, you are ensuring that all of the logical disk pieces on that physical disk are known to the system. The following menu is displayed when you select option 1 from the Physical Disk Management Menu.

```
           Physical Disk Registration Menu

  1. Register a Physical Disk
  2. Deregister a Physical Disk
  3. List Registered Physical Disks

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

  Enter choice:
```

Register a Physical Disk

As an example, let's add disk cied(0,0) to the system. The **diskman** program queries you for a physical disk specification and tries to register it.

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

Physical Disk cied(0,0) has been registered.

Press New Line when ready to continue.
```

If the disk is already registered, your screen will display:

```
        Physical Disk cied(0,0) is already registered.
```

If the registration fails:

```
        Physical Disk cied(0,0) could not be registered.
```

Deregister a Physical Disk

As an example, let's remove disk cied(0,1) from the system. The **diskman** program queries you for a physical disk specification and tries to deregister it.

```
        Enter the Physical Disk specification in DG/UX common
        format: cied(0,1) ↵

        Physical Disk cied(0,1) has been deregistered.

        Press New Line when ready to continue.
```

If the disk is not registered, the system displays:

```
        Physical Disk cied(0,1) is not registered.
```

If the deregistration fails:

```
        Physical Disk cied(0,1) could not be deregistered.
```

List Registered Physical Disks

This option displays the currently registered physical disks on the system.

```
       Currently registered physical disks are:

          cied@18(FFFFEF00)
          cied@19(FFFFF100)

        Press New Line when ready to continue.
```

If no disks are registered:

```
        There are currently no Registered Physical Disks.
```

Procedure 7.1.2: Add, Recover, or Display Bad Blocks on a Physical Disk

Purpose	To locate bad blocks and add them to the bad block table. To remove blocks from the bad block table that are no longer bad.
diskman menu	Bad Block Management Menu
References	**diskman(1M)**

When you select number 2 from the Physical Disk Management menu, the following is displayed:

```
           Bad Block Management Menu

  1. Add Bad Blocks to a Physical Disk's Bad Block Table
  2. Recover Bad Blocks from a Physical Disk's Bad Block Table
  3. Display a Physical Disk's Bad Block Table

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

  Enter Choice:
```

These selections are used as explained in the following sections.

Add Bad Blocks to a Physical Disk's Bad Block Table

The bad block table is a place on your disk that is created when you format the physical disk. Bad blocks are 512-byte sections of the physical disk that may be unreliable for storage and retrieval of information.

There may be other bad blocks besides the ones listed in the bad block table. For instance, a bad block, say 55555, may have gone bad since surface analysis time. If you suspect a block is unreliable or diagnostics has shown a block to be unreliable, you can add that block to the bad block table by hand with this selection. You will be asked to specify the physical disk and the physical disk address of the bad block that you want to add to the table. You are queried:

```
Enter the Physical Disk specification in DG/UX common
format: cied(0,1) ↵
```

```
The Physical Disk must be registered for this operation.
Do you want to register it? [y] ↵

Enter the Physical Disk addresses of bad blocks, one per line.
Use New Line to terminate the list.
```

Hit the New Line key after each address you enter. The second statement is repeated until you press New Line without entering a bad block address. Below, let's enter two addresses:

```
Enter the Physical Disk Address: 55555 ↵
Enter the Physical Disk Address: 44444 ↵
Enter the Physical Disk Address:         ↵
```

If any of the bad blocks are readable (not bad according to the system), you are informed about them and asked if you still want to add those blocks.

```
The following blocks are readable:

44444

Do you want to add them to the Bad Block Table anyway? [n] ↵

The following blocks were added to the Bad Block Table:

55555

Press New Line when ready to continue.
```

If no good blocks exist to which to map bad blocks, the system prints an error message telling you that it cannot add the bad blocks.

Recover Bad Blocks from a Physical Disk's Bad Block Table

This option recovers the remapped blocks which 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. Later, let's say you've been having a lot of I/O errors so you have your disk serviced. After servicing, you want to remove block 100000 and all other "used-to-be" bad blocks from the bad block table. When you choose this selection, the system responds as follows:

```
Enter the Physical Disk specification in DG/UX common
format: cied(0,1) ↵

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ↵

Beginning Bad Block recovery...
```

As each block is recovered, a message is printed:

```
Recovered block at Physical Disk Address xxxxxx.
Recovered block at Physical Disk Address xxxxxx.

xx bad blocks were recovered on Physical Disk cied(0,1).

Press New Line when ready to continue.
```

Display Physical Disk's Bad Block Table

After you have formatted a disk, use this option to display and remap bad blocks (if any). If you supply the disk drive specification, the **diskman** program will list the addresses (in decimal form) where bad blocks are located.

When you choose this selection, the system responds as follows:

```
Enter the Physical Disk specification in DG/UX common
format: cied(0,1) ↵

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ↵

Bad Blocks exist at the following Physical Disk Addresses:

433560
466000
466660

Press New Line when ready to continue.
```

Procedure 7.1.3: Display a Physical Disk's Layout

Purpose	To display a table showing how areas on a given physical disk are being used.
diskman menu	Physical Disk Management Menu
References	**diskman(1M)**

When you want to know the layout of your physical disks for planning, inventory, or other purposes, choose this option. It displays the information on the sections of a physical disk: sizes, logical disk pieces, and physical addresses. It will also print the total disk capacity and free space on a specified disk in blocks.

The Primary System Area (PSA), is created when you format the physical disk. It contains the initial bootstrap program and information describing the layout of the physical disk. You will be asked for the disk drive specification in DG/UX common format. Then, the system responds as follows:

```
         System Areas on Physical Disk cied(0,0):


          Area Name           LD Piece     Physical Disk    Size
                              Number       Address          in
                                           of Area          Blocks

    Primary System Area       ....          0                8
    System Bootstrap Area     ....          8                64
    Secondary System Area     ....          72               8
    Primary Bad Block Table   ....          80               1
    Primary LDP Table         ....          81               9
    Bad Block Remap Area      ....          90               84
    Secondary Bad Block Table ....          174              1
    Secondary LDP Table       ....          175              9
    thor                      1 of 1        180              20000
    comm                      2 of 2        182              10000
    (Free Space)              ....          184              331472

Total Physical Disk Size: 371640 blocks.
Unallocated Space: 331472 blocks.


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

For logical disks, we see that piece 1 of **thor** and piece 2 of **comm** are located on physical disk cied(0,0).

Procedure 7.1.4: Display a Physical Disk's Label

Purpose	To display a physical disk's geometry.
diskman menu	Physical Disk Management Menu
References	**diskman(1M)**

When you select this function, the system responds as follows:

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

    The Physical Disk must be registered for this operation.
    Do you want to register it? [y]

    Disk Label for physical disk cied(0,0):

Total cylinders:                   1224   interleave:                1
OS visible cylinders:              1220   head skew:                 3
Tracks per cylinder:                 15   Cylinder skew:             6
Sectors per track:                   35   Head group skew:           0
Bytes per unformatted sector:       512   Spares per track:          1
Bytes per logical sector:           512   Byte per data preamble:   16
Bytes in mfg's defect info:           0   Bytes per id preamble:    14
Mgs's defect area start sector:       0   Base head for volume:      0
Number of relocation areas:           0   Bytes in gap 1:           12
Sectors per relocation area:          0   Bytes in gap 2:           13
SMD extended addressing is not used.
A final short sector does not exist.
```

Procedure 7.1.5: Format a Physical Disk

Purpose	To format a physical disk.
diskman menu	Physical Disk Management Menu
References	**diskman(1M)**

When you select this option from the Physical Disk Management menu, the following is displayed:

```
          Physical Disk Formatting Menu

    1. Install a Disk Label on a Physical Disk
    2. Perform Hardware Formatting on a Physical Disk
    3. Create DG/UX System Areas on a Physical Disk
    4. Install Bootstraps on a Physical Disk
    5. Perform Surface Analysis on a Physical Disk
    6. All of the above

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

    Enter Choice:
```

Note that item 1 on the menu will only be displayed when you're using stand-alone **diskman**. We recommend that you select item 6 to perform all steps.

Install a Disk Label on a Physical Disk

Use this function to install a particular disk label on a physical disk. A disk label contains the disk geometry (i.e tracks per cylinder, bytes per sector, etc.) Every disk must contain this information so that it can be accessed by the operating system. This function is generally used in the default case, choice 6. The interaction is:

```
Enter Choice: 6 ↵
Enter the Physical Disk specification in DG/UX
common format: cied(0,2) ↵

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ↵

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

If the disk already has a label, you will be asked if you want to reinstall it.

```
A label already exists on this physical disk.
Do you want to reinstall the label? [n]
```

If a label exists and you opt to reinstall it, information is displayed as follows:

```
   Disk    Model#      Size

1. ESDI    123456      380 MB
2. ESDI    123456      766 MB
3. SCSI    123456      766 MB
4. None of the Above

What type of disk do you have?
```

If you select one of the numbers above, that disk label will be installed. If you select "None of the Above," you will need to enter 22 parameters for your disk. You will get the following prompts:

```
Enter the total cylinders per drive:
Enter the OS visible cylinders per drive:
Enter the tracks per cylinder:
Enter the sectors per track:
Enter the bytes per logical sector:
Enter the bytes in mfg defect information:
Enter the bytes per unformatted sector:
Enter mfg defect information start sector:
Enter the number of relocation areas:
Enter the sectors per relocation area:
Enter the interleave:
Enter the head skew:
Enter the cylinder skew:
Enter the head group skew:
Enter the spares per track:
Enter the bytes per data preamble:
Enter the bytes per id preamble:
Enter the base head for volume:
Enter the bytes in gap 1:
Enter the bytes in gap 2:
Does the drive use SMD extended addressing?
Does the drive have a final short sector?
```

If the install was successful:

```
Disk Label has been installed.
```

Perform Hardware Formatting on a Physical Disk

Use this option if you need to hard-format a physical disk. Hardware formatting differs from surface analysis. Hardware formatting marks each sector, track, etc. of the disk so the controller can read it. This is usually done by the manufacturer. If you have a disk that is not already hardware formatted, you need to use this option.

The interaction is as follows:

```
The Physical Disk must be deregistered for this operation.
Do you want to deregister it? [y] ↵

All Data General disks, and most other manufacturer's disks
are already hardware formatted and do not need to be reformatted.
See the disk's manual before doing this.
Do you wish to have the disk hardware formatted? [n] ↵
```

Create DG/UX System Areas on a Physical Disk

The DG/UX operating system needs system areas to describe the file systems on a disk. There is a Primary System Area (PSA) and a Logical Disk Piece Table that describes the logical disks that are on the physical disk. Another system area is the Bad Block Table which the OS uses to remap bad blocks.

The interaction is:

```
The Physical Disk must be deregistered for this operation.
Do you want to deregister it? [y] ↵

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

WARNING: this operation will destroy any data on the
physical disk cied(0,2).

Do you want to continue? [y] ↵

The physical disk cied(0,2) is nnnnnnn blocks in size.
Enter the number of blocks to allocate for the remap
area: [189] ↵
```

Install Bootstraps on a Physical Disk

The **diskman** program contains low-level bootstrap programs used to the DG/UX system image. These programs are written to disk by **diskman** when you format the physical disk. If you are adding a new release of the DG/UX system or the contents of your disk have been destroyed, you will need to reinstall the bootstraps.

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

  The Physical Disk must be deregistered for this operation.
  Do you want to deregister it? [y] ↵

 Installed Bootstraps on the Physical Disk cied(0,2).
```

Perform Surface Analysis on a Physical Disk

If this disk is on a controller that performs hardware bad block remapping, you will be informed and asked if surface analysis is still desired. The system displays:

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

Do you want to perform surface analysis on this Physical
Disk? [y] ↵
```

If you run surface analysis, we recommend that you run *all* test patterns. The process takes about 20 minutes per 100 Mbytes and will depend upon your physical disk model and CPU. An average is about an hour. Before surface analysis begins, you are queried:

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

Do you want to run all the test patterns? [y] ↵
The Physical Disk cied(0,0) is 100000 blocks in size.
```

As surface anaylsis proceeds, the system displays:

```
Beginning Surface Analysis...
Surface analysis is 27 percent complete; finished in 13 minutes.
Surface analysis is 54 percent complete; finished in  8 minutes.
Surface analysis is 81 percent complete; finished in  3 minutes.

Surface analysis finished
xx bad blocks were found and remapped.

The Physical Disk cied(0,2) has been formatted.
Do you want to register it? [y] ↵
The Physical Disk cied(0,2) has been registered.
```

Procedure 7.2: Logical Disk Management

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.

You can name logical disks anything you want as long as the names are 1 to 32 characters long. The characters must be among a-z, A-Z, 0-9, - (hyphen), _ (underscore), or . (period).

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

When you create a logical disk, you must create all pieces of that logical disk. This means it is important to plan ahead. For instance, if you intend to have a logical disk that spans three physical disks, then when you create the logical disk you must create each piece on a specified physical disk. For instance, you could create logical disk **tax_86** with pieces on disk 0, disk 1, and disk 3, only if you do it all at once at creation time.

You cannot add pieces after you've created a logical disk. If one piece is damaged or inaccessible, you must delete all remaining pieces before you can re-use the associated physical disk space.

The Logical Disk Management menu is displayed when you select option 2 from the main menu:

```
        Logical Disk Management Menu

  1. Create a Logical Disk
  2. Delete a Logical Disk
  3. Display Information about a Logical Disk
  4. Copy a Logical Disk
  5. Display Information about a Logical Disk Piece
  6. Delete a Piece of a Damaged Logical Disk

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

  Enter Choice:
```

Procedure 7.2.1: Create a Logical Disk

Purpose	To create logical disks.
diskman menu	Logical Disk Management Menu
Note	You must create all pieces of a logical disk when you create the logical disk. You cannot add pieces later.
References	**diskman(1M)**

If you have finished formatting your physical disk, you are ready to create one or more logical disks. If you have not formatted your physical disk, go back to the Physical Disk Management Menu.

When you create a logical disk, the system assumes you are starting with piece 1; it is your option to create a possible eight pieces under a single logical disk name. So when you create **comm**, the **diskman** program views it as piece number 1 of **comm**.

```
Enter the Logical Disk name: comm
```

You will be prompted for information on each piece (of a possible 8 pieces) that will be part of this logical disk. Below, we'll create a logical disk named **comm** that consists of one piece on physical disk 0.

```
Logical Disk Piece 1:

Enter Physical Disk specification in DG/UX common
format: cied(0,0) ⤶

Do you want to display the layout of this Physical Disk? [y]
```

Press New Line if you want to display the layout of the physical disk. The logical disk will begin at the first available location on the physical disk if you select the default.

```
Enter the Physical Disk Address of the starting block
of Logical Disk Piece 1: [default] ⤶

Enter the size in blocks of Logical Disk Piece 1: [default] ⤶

Do you want to specify any more Logical Disk Pieces? [n] ⤶
```

If you specify more pieces, **diskman** displays the following message and begins the create loop again.

```
You have allocated n blocks so far for this Logical Disk.
```

But if you're finished creating pieces, you will get either a successfull or an unsuccessful report:

```
        The Logical Disk "comm" has been created.
```

or

```
        Could not create the Logical Disk "comm".
```

If the creation was successful, you will be asked if you want to make a file system on the new logical disk:

```
        Do you want to make a file system on this logical disk? [y] ↵

        Enter the Logical Disk name: [comm] ↵
```

This procedure uses the **mkfs(1M)** program to make a file system. Your next query is:

```
       WARNING: this operation will DESTROY the contents of the
       Logical Disk "comm".
       Do you want to continue? [y] ↵


       Enter the additional mkfs flags and options you wish
       to specify: ↵
```

No additional information is required, but you may specify additional mkfs flags and options if you wish. We pressed New Line because **mkfs** automatically sets file system characteristics. At the default, the system displays:

```
        Made a File System on the Logical Disk "comm".
```

Procedure 7.2.2: Delete a Logical Disk

Purpose	To delete all pieces that make up a currently usable logical disk. All pieces of the disk must be intact to use this procedure.
diskman menu	Logical Disk Management Menu
References	**diskman(1M)**

As you organize and reorganize your disk usage over time, you'll probably find it necessary to delete a logical disk. Below, we'll delete a logical disk that is spread over three physical disks. Note that you can only delete logical disk **accounts_86** if all pieces are intact. This means that all pieces of a logical disk must be on physical disks that are in service. If one of those physical disks has suffered a failure, then any logical disks associated with that disk cannot be deleted with this procedure. Procedure 7.2.6 handles the case of recovering physical disk space by deleting pieces of logical disks that are on damaged physical disks.

Below, let's assume that we want to delete a functional logical disk:

```
        Enter the Logical Disk name: accounts_86 ⤶

      The Logical Disk accounts_86 consists of the following pieces:


Piece        Physical Disk          Starting Physical       Size in
                                    Disk Address            Blocks


1              cied(0,0)                  3565                2000
2              cied(0,2)                  48876               1500
3              cied(0,3)                  716                 1000

        Do you want to delete the Logical Disk "accounts_86"? [y] ⤶

        The Logical Disk "accounts_86" has been deleted.

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

If you try to delete a logical disk when one of its pieces is on a damaged physical disk, the system displays:

```
        Could not remove the Logical Disk "accounts_86".
```

Procedure 7.2.3: Display Information About a Logical Disk

Purpose	To get an overall view of all logical disks in service.
diskman menu	Logical Disk Management Menu
References	**diskman(1M)**

This option lists everything that the system knows about a logical disk. If you intend to delete or change pieces of logical disks, you need to know where all the pieces are located and how much space in blocks is being used by each logical disk piece.

```
        Enter the Logical Disk name: comm ↵
```

After you give the name of the logical disk, the system responds as follows:

```
        The Logical Disk "comm" consists of the following pieces:

Piece       Physical Disk          Starting Physical         Size in
                                   Disk Address              Blocks

1             cied(0,0)                 376542                 3000
2             cied(0,1)                 218576                 2000

        Size of Logical Disk: 5000 blocks

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

Procedure 7.2.4: Copy a Logical Disk

Purpose	To make a backup copy of a logical disk on another logical disk.
diskman menu	Logical Disk Management Menu
References	**diskman(1M)**

Although you have the option of using the **dump(1M)** command to write logical disk information to tape, copying that information to another logical disk is much faster. Below, "source" is the disk you wish to duplicate. Before you can make the backup copy, you must first create a logical disk of exactly the same size as the one that will be copied. This option is available in both versions of **diskman**.

```
Enter the Source Logical Disk Name: assets ↵

Enter the Destination Logical Disk Name: bkup_assets ↵

WARNING: this operation will DESTROY any data currently
on the Logical Disk "bkup_assets".

Do you want to continue? [y] ↵

Beginning  Logical Disk copy...

The Logical Disk "assets" has been copied to the Logical
Disk "bkup_assets".

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

Procedure 7.2.5: Display Information about a Logical Disk Piece

Purpose	To list the addresses and sizes for specified logical disk pieces.
diskman menu	Logical Disk Management Menu
References	**diskman(1M)**

This option displays information, one piece at a time, for each of the eight pieces that can make up a logical disk. This option is available in both versions of **diskman**.

Below, we seek information on piece number 2 of logical disk **comm**.

```
Enter the Logical Disk Name: comm ↵

Enter the Logical Disk Piece Number: 1 ↵

This Logical Disk Piece is on cied(0,0).
Its starting Physical Disk Address is block 218576.
Its size is 2000 blocks.

Press New Line when ready to continue.
```

Procedure 7.2.6: Delete a Piece of a Damaged Logical Disk

Purpose	To recover physical disk space associated with out-of-service logical disk pieces.
diskman menu	Logical Disk Management Menu
Caution	Use this procedure only for deleting pieces associated with damaged physical disks. If you try to delete a piece of a usable logical disk, you will make that entire disk inaccessible. If you wish to delete an entire usable logical disk, see Procedure 7.2.2.
References	**diskman(1M)**

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.

This procedure allows you to recover physical disk space so that you can reformat it and use it for other logical disks. This is useful when you have a logical disk spanning two or more physical disks. For instance, say you have a two-piece logical disk named **trimble**. One piece of **trimble** is on unit 0 and the second piece is on unit 1. Suppose the head scratches unit 1, putting it out of service. You can no longer use **trimble** because one of its pieces is on damaged physical disk 1. All other space associated with **trimble** on any other physical disks is also unusable. If you want to use that space, you must first recover the space by deleting the associated piece or pieces of **trimble**.

Below, we'll delete the surviving piece of **trimble** on unit 0:

```
WARNING: this operation will DESTROY the contents of a
Logical Disk.  Use this to recover the space being
used by the surviving pieces of an already-damaged
Logical Disk.

Enter the Physical Disk specification in DG/UX common
format: cied(0,0)

Enter the Logical Disk Name: trimble ⏎

Enter the Logical Disk Piece Number: 1⏎

This Logical Disk Piece is on the Physical Disk cied(0,0).
Its starting Physical Disk Address is block 218576.
Its size is 20000 blocks.

Do you want to delete this Logical Disk Piece? [y] ⏎
```

```
        Piece 1 of the Logical Disk "trimble" has been deleted.

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

Now we can re-use the vacant space on physical disk cied(0,0).

Procedure 7.3: File System Management

Use the selections from this menu to create and check file systems. File systems are created via the **mkfs(1M)** program and checked via the **fsck(1M)** program.

The following menu is displayed when you select option 3 from the **diskman** Main Menu:

```
          File System Management Menu

     1. Make a File System
     2. Check a File System

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

Enter Choice:
```

Procedure 7.3.1: Make a File System

Purpose	To make a file system on a logical disk.
diskman menu	File System Management Menu
Caution	This command erases all information on the logical disk.
References	**mkfs(1M)**

Use this procedure to create a new, empty file system on a logical disk. After creating the file system, you will have to mount it on a directory (**sysadm mountfsys**) before users can access it. We recommend that you use the logical disk name associated with the file system when you mount the file system on a directory. For instance, if you have a file system on logical disk **comm**, then you should create a mount directory named **/comm**. Then, after you mount the file system, its name would be **/comm**.

Type ? for HELP if you are unsure how to use this option. Refer to **mkfs(1M)** for more information.

```
Enter the Logical Disk Name: comm ↵

WARNING: this operation will DESTROY the contents of the
Logical Disk "comm".
Do you want to continue? [y] ↵


Enter the additional mkfs flags and options you wish
to specify: ↵

Made a File System on Logical Disk "comm".
```

Notice that above we typed only a logical disk name, no options. For most cases, this will suffice for creating your file systems. That is, file system characteristics such as inode density (potential number of files), region size, free space, etc., are automatically set by **mkfs**. If you have special file system requirements, see **mkfs(1M)** for details on changing defaults.

Procedure 7.3.2: Check a File System

Purpose	To run the **fsck** check and repair program on a file system.
diskman menu	File System Management Menu
References	**fsck(1M)**, Appendix D

Use this option to check a file system for inconsistencies. The **fsck** program checks blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region (DAR) information. **Fsck** reports any inconsistencies; it is your option to fix or ignore them. For a detailed discussion of **fsck**, see Appendix D in this manual.

The following prompt is displayed:

```
 Enter the Logical Disk Name: comm ↵

 Enter the additional fsck flags and options you wish
to specify:   -p ↵
```

The example above runs **fsck** with the **-p** option on file system **/comm**. The **fsck** program will display messages about the success or failure of the check.

Command Line Options

As you become comfortable with disk operations, you may choose to execute some **diskman** functions from the shell. You can invoke stand-among **diskman** with command line options that perform functions without going through the menus. Or you invoke it via **sysadm diskmgmt**. Here are some options you may find useful. We list the functions provided as command line options below:

- To display all registered disks, type:

  # **diskman list_reg**

- To register a physical disk, type:

  # **diskman reg_disk** *PD*

  where PD is the physical disk specification in DG/UX common format.

- To deregister the physical disk, type:

  # **diskman dereg_disk** *PD*

  where PD is the physical disk specification in DG/UX common format.

- To display logical disk information, type:

  # **diskman ld_info** *LD*

  where LD is the logical disk identifier.

Note that you will have to surround the physical disk specification with quotes since parentheses are shell metacharacters.

End of Chapter

Chapter 8
File System Management

Operations involving creating file systems and logical disks are done on machines that have their own physical disks. Other operations, such as mounting, unmounting, and modifying file system tables are done on machines with or without physical disks.

This chapter gives brief explanations of logical disks, mounting file systems, dump cycles, and shows you how to manage your DG/UX file systems. You'll use **sysadm** to add, delete, mount, unmount, restore file systems, and to make backup tapes. To create file systems, use **sysadm diskmgmt**. The major sections of this chapter are:

- File System Terms
- File System Perspectives
- Making File Systems Accessible
- File System Management Procedures
- Expert File System Information

File System Terms

You may want to review the information on file systems and logical disks in Chapter 2 and Chapter 7. Chapter 4 contains file system performance information. We use the following terms in this chapter:

/etc/fstab The **fstab**(4) file describes local and remote file systems available to the local machine. File systems must be listed in **fstab** to be accessible. This file is read by commands that mount, unmount, dump, restore, and check file systems. The entries in **fstab** are accessed by the routines in **getmntent**(3).

mount point directory

After you name a logical disk (**thor**) and format the logical disk, you then have a file system. Next you mount the file system on a directory named **/thor**. From now on, the new file system carries the name of the directory on which it was

	mounted. In **sysadm**, mount directory name = file system name.
mount	To attach a file system to a directory, making it accessible to users.
unmount	To detach a file system from a directory, making it inaccessible to users.
initialize	To make (**mkfs(1M)**) a file system on a logical disk. Initialization clears or erases anything in the specified disk region. Do this with **diskman**.
dump cycle list	A scheme for doing daily, weekly, and monthly backups on tape. A default dump cycle list is supplied with the DG/UX system.
pass number	A number from a file system's **fstab** entry that indicates the order in which the **fsck** program will check the file system.
read-write mode	Access permission that allows general use of a file system. Specified by **rw**. "Read only" is specified by **ro**.
NFS	Network File System. A separate software package that allows you to access remote file systems as though the remote file systems existed on your machine. For more information, see *Managing NFS and its Facilities on the DG/UX™ System*.

File System Perspectives

As discussed in Chapter 7, the DG/UX system uses logical disks, a feature that provides more flexibility for organizing file systems. The operating system treats a logical disk as if it were a real physical disk.

You can view file systems in two ways. From the operating system's perspective, a file system is associated with a logical disk, which in turn is associated with sections of physical disks accessed through a device node. From the user's perspective, a file system is a hierarchical directory structure. Logical disk names assigned by a user are incorporated into the node names used by the operating system. The association between a directory name at the user level and a node name at the system level is where the user's view meets the system's view.

The kernel creates device nodes at boot time. These device nodes provide the operating system with access to logical disks. For each device and for each logical disk, the kernel creates a device node automatically each time you boot the system.

The Operating System's View of File Systems

From the operating system's point of view, a file system is associated with sections on one or more physical disks. Logical disks form the bridge between file systems and physical disks. The file system associated with the logical disk is mounted (made available to users) in the directory structure.

You can think of the relationship between file systems, logical disks, and physical disks as a three-level hierarchy.

- **The file system** is the level at which the user interacts with the system.
- **The logical disk** is the intermediate level that associates the file system with the physical disk. The file system and the rest of the operating system interact at this level.
- **The physical disk** is the level at which the operating system interacts with the hardware.

Logical disks have other functions besides acting as bridges between physical disks and file systems. You received a preloaded **swap** logical disk or you created one during installation. This logical disk will not be associated with, nor does it look or act like, a file system. It is an area of a disk that is only accessed directly 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, the world appears to be a single hierarchical directory structure. If you mount a new file system, the user's world 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 world disappears. The system prevents you from inadvertently unmounting a file system if a user is in a directory of that file system. For example, if a user is in **/usr/lib**, you would be warned if you tried to unmount the **/usr** file system from the **/** (root) file system.

Creating a File System

You can access **diskman** to create file systems in two ways. You can invoke stand-alone **diskman** from the SCM by typing

```
SCM> b cied()usr:/stand/diskman ↲
```

To create file systems after the operating system is running, use the stand-among version of **diskman** by invoking **sysadm diskmgmt**.

Making File Systems Accessible

A file system starts out as a set of disk blocks in a logical disk. You use **diskman** to allocate space (create one or more *pieces*) on a physical disk and you assign that allocated space a name, say **thor**. Thus, you create a logical disk named **thor**. Next, with **diskman**, you format the disk blocks in **thor** into a file system structure of directories, files, filenames, and file modes. But, even though you may have created a file system with **diskman**, that file system is not yet accessible to users. In this chapter, you will make file systems accessible by attaching or *mounting* them on a directory of your choice called a *mount directory*. Once mounted, the file system officially has a name; by convention, we give the file system the same name as the directory on which it is mounted. And remember, the mount directory must itself be mounted before you can attach anything to it. For convenience, we give the mount directory the same name as the logical disk. So, for logical disk **thor**, we have mount directory (and file system name) **/thor**.

Finally, all file systems must be listed in **/etc/fstab** before they are accessible to users. When you use **sysadm addfsys**, you will automatically add file systems to **/etc/fstab**. For more information on this file, see **fstab(4)**.

Every time you boot your system, the kernel creates new device nodes, therefore, in our example above, the node name **/dev/dsk/thor** will be created.

The following figure shows mount directory **/thor** branching from **/usr**. We now have a new file system named **/usr/thor**.

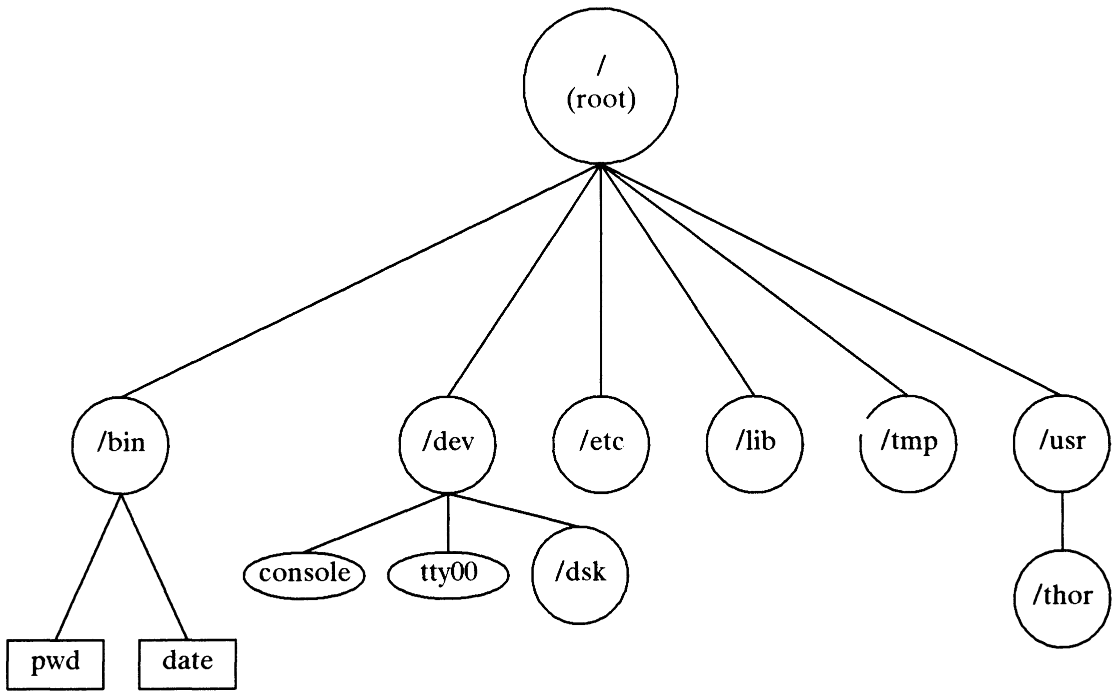


Figure 8-1 Mounting a File System

You can think of each file system as an independent directory tree. The **mount** command "grafts" a file system tree onto a larger tree. The top of the total tree is the **/** (root) directory, which is also the top directory of the root file system.

So, after you have used **diskman** to create a logical disk and make a file system on the logical disk, you must create a mount directory, and mount the file system on that mount directory. You can do this with the **addfsys** command in Procedure 8.1: Add a File System Entry. When you conclude with **addfsys**, the file system will be accessible to users.

The Default File System Dump Cycle

As a part of routine management, you or someone else will make backup tapes of your file systems. We have included a default dump cycle method designed around daily, weekly, and monthly backups. With this method, you schedule according to the dump cycle list. A program called **fsdump** reads the information from this list and interacts with you to perform each dump. Procedure 8.7 shows you how to use the default dump cycle and how to change it to fit your needs.

File System Management Procedures

This section describes the procedures you will use to manage your file systems. When you select **fsmgmt** from the **sysadm** Main Menu, the following is displayed:

```
                         File System Management

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

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

Procedure 8.1: Add a File System Entry

Purpose	To add an entry to the table of file systems in **/etc/fstab**. Also creates mount directory and mounts the file system if desired.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**addfsys**
Note	The **sysadm** program knows only about file systems that are listed in **/etc/fstab**.
References	**fstab(4)**

For this procedure, supply the name of the mount directory and name of the logical disk associated with the file system. If these are already in **/etc/fstab**, an error message is printed, and **addfsys** exits. If not, then you will be queried for the file system type, the read-write mode, the NFS mount mode (hard or soft), the dump cycle, and the **fsck** pass number. When you select **addfsys**, the system responds as follows:

```
Mount directory name? /usr/thor ↵

Is this a local file system? [yes]
```

At this point, we will split the **addfsys** dialogue into two examples: one for local file systems, and another for remote file systems, those accessed via NFS.

Local File Systems

If you answer yes to the last query, the system responds as follows:

```
Logical disk name? thor ↵

Writeable? [yes] ↵
```

Answering **yes** to `Writeable` allows users to create or modify files on this file system. If you answer *no*, then users will have only read permission for this file system, not write or execute permissions.

The next query, `Dump cycle?`, refers to how often this file system should be backed up on tape, or archived. Indicating a dump cycle choice does not mean that archives (making backup tapes) will occur automatically. This information is collected and used later by **fsdump** when you are ready to do an archive. The choices 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.

See Procedure 8.9 for more explanation of d, w, and m.

```
Dump Cycle? [d] ↵

fsck Pass Number? [1] ↵
Export? [yes] ↵

Mount the file system? [yes] ↵
The file system has been mounted.

Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

The **fsck** pass number indicates the pass on which an **fsck -p** process should check this file system. This number is a digit between 0 and 9. File systems with pass numbers between (and including) 1 and 9 will be checked in order. That is, all file systems with number 1 are checked first, then those with number 2, and so on. 0 indicates that a file system should never be checked. For more on **fsck**, see Appendix D.

After the **fsck** number, we hit New Line for the yes default on exporting this file system. When a file system is exported, it can be mounted and used by other computer systems.

Remote File Systems

Let's assume we have NFS and want to add a remote file system. The remote system is **sys5**. The mount directory is **/comm/prog**. We got this information from the system administrator of **sys5**.

```
Is this a local file system? [yes] n ↵

Remote host name? sys5 ↵
Remote mount directory? /comm/prog ↵
Writeable? [yes] ↵
Hard mount? [yes] ↵
```

If an NFS file system is hard mounted, user processes will wait indefinitely for file system accesses to be completed. This means that if the remote machine on which the file system resides is not responding (because it is down, for example), user programs will appear to hang. If an NFS file system is soft mounted, user processes will receive an I/O error if the remote file system does not respond. Answering **no** gives a soft mount.

If the directory you specified as a mount directory does not exist, the system displays:

```
        The directory /comm/prog does not exist.
        Create /comm/prog? [yes]
```

So you can create the directory now if needed.

```
        Mount the file system? [yes] ↵
        The file system has been mounted.

        Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

Users cannot access the file system unless it is mounted. If you do not mount the file system now, it will be automatically mounted when you reboot.

Procedure 8.2: Delete a File System

Purpose	To delete an entry from the table of available file systems in **/etc/fstab**.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**delfsys**
Note	You may delete only one file system at a time.
References	**fstab(4)**

You must specify the mount directory name. The **delfsys** command deletes the entry in **/etc/fstab** for the file system you specify. With the entry deleted, that file system can no longer be mounted, and therefore is no longer accessible to users. All information is still on disk, so if you re-add the entry, it becomes accessible again.

If you select **delfsys**, the system responds as follows:

```
Mount directory name? /usr/thor ⮐

Do you really wish to delete this file system entry? [no] ⮐

Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

Answer **y** to delete the file system. If you want only to disable the file system, exit with **q** and select **modfsys**.

Procedure 8.3: List File Systems

Purpose	To list the entries in **/etc/fstab**.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**lsfsys**
References	**fstab(4)**

For each file system in **/etc/fstab**, this command shows the logical disk name, the mount point directory, the file system type, the access mode, the NFS mount type, the dump cycle, and the **fsck** pass number. If you select **lsfsys**, the system responds as follows:

```
Logical            Mount            FS          NFS     Dump    Fsck
Disk               Directory        Type   RW   Mount   Cycle   Pass
---------------   -------------     -----  --   -----   -----   -----
/dev/dsk/root       /               local  rw             w       1
/dev/dsk/swap       [swap]          swap                  x       0
/dev/dsk/dBase      /dBase          local  rw             d       3
/dev/dsk/thor       /usr/thor       local  rw             d       2
sys5:/dev/dsk/sys   /comm/prog      remote rw    soft     x       0


       Press New Line to see the fsmgmt menu [?, ^, q]:
```

Procedure 8.4: Modify a File System Entry

Purpose	To modify an entry in the **/etc/fstab** file system table. Change logical disk name, file system type, access mode, dump cycle, and **fsck** pass number.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**modfsys**
Note	A file system must be unmounted before you can modify it.
References	**fstab(4)**

The **modfsys** command locates the specified file system entry and uses its values as defaults for the queries below. If no entry exists, **modfsys** exits with an error message. If you select **modfsys**, the system responds as follows:

```
Mount directory name? /systems

Is this a local file system? [yes] ↵
```

Above, we defaulted to **yes** because the file system is local, that is, it exists on a physical disk connected to our processor. For a remote file system accessed via NFS, enter **n** for no. At this point, we will split the **modfsys** dialogue into two examples: one for local file systems, and another for remote file systems.

Local File Systems

We'll begin with the logical disk name:

```
Logical Disk Name? [systems] ↵

Writeable? [yes] n
```

Above, we changed the permissions on **/systems** to be "read only." Below, we continue to press New Line to select the defaults. Notice that we are exporting this local file system. Once exported, it can be mounted and used by other systems.

```
Dump cycle? [m] ↵
fsck pass? [1] ↵
Export? [yes] ↵
```

```
The entry for /systems has been modified.

Mount the file system? [yes] ↵

Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

If you do not mount the file system, no one can access it until you reboot your system. Rebooting causes an automatic mounting of file systems.

Remote File Systems

Let's go back to the local file system question. Let's say we want to modify file system **/comm/prog** on remote host **sys5**:

```
Is this a local file system? [yes] n ↵

Remote host name? sys5 ↵
Remote Mount Directory? /comm/prog ↵
Writeable? [yes] ↵
Hard Mount? [yes] ↵

The entry for sys5:/comm/prog has been modified.

Mount the file system? [yes] ↵

Press New Line to see the fsmgmt menu [?, ^, q]:
```

Procedure 8.5: Mount a File System

Purpose	To mount a file system on a directory so that users may access the file system.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**mountfsys**
References	**fstab(4), mount(1M)**

You will normally mount file systems for the first time when you add them with the **addfsys** command. File systems are automatically mounted when the system changes to run levels 2 or 3, but there are times when you might want to mount a file system manually.

In this procedure, you must supply the mount directory name. That name must exist in **/etc/fstab**. If it does not, exit with **q** and use **addfsys** to make the entry. When you select **mountfsys**, the system responds as follows:

```
Mount directory Name? /usr/thor ⮐
```

Above, we responded with a specific mount directory. You can also respond to this first query with the following:

list	Lists all mount directory names.
all	Mounts all file systems.
remote	Mounts all remote file systems.
local	Mounts all local file systems.

If the mount directory does not exist, you can create it now:

```
Create /usr/thor? [yes]
```

But, if the **mountfsys** command succeeds, the system displays

```
File system /usr/thor has been mounted.

Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

Procedure 8.6: Unmount a File System

Purpose	To unmount a file system on a directory, making it inaccessible to users.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**unmountfsys**
References	**fstab(4), mount(1M)**

Use this procedure when you want to unmount file systems. Our example below unmounts a specific file system. Another way to use this command is by responding to the first query with one of the following:

list	Lists all mount directory names.
all	Unmounts all file systems.
remote	Unmounts all remote file systems.
local	Unmounts all local file systems.

Below, we'll unmount a file system called **/accounts** (which is also the name of the file system's mount directory).

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

```
Mount directory Name? /accounts ↵
```

If the file system is not mounted, the following is displayed:

```
File system /accounts is not mounted.
```

If the unmount succeeds, the following is displayed:

```
File system /accounts has been unmounted.
```

Or, if the unmount fails:

```
An error has occurred using the unmount command.  The error
message is: <error_msg>

Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

Procedure 8.7: Modify Dump Cycle

Purpose	To modify the current position in the dump cycle.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**modcycle**
References	**dump2**(1M)

This command changes the current position of the pointer in the dump cycle list. This list contains one entry for each day of the month. The pointer indicates the current day's entry to **fsdump**, which performs the dump. Each entry shows the dump cycle letters that match the dump for the day (this selects the file systems to be dumped) and the tape label information to be used for the day's dump tapes. When a dump is completed, the pointer is automatically moved to the next day's entry. On the first day of the month, the pointer is reset to the top of the list to restart the dump cycle.

It's possible for the current pointer position in the list to be wrong if dumps are skipped for a day or more. With **modcycle**, you can manually move the pointer to the desired line in the list.

When you select the **modcycle** command, the system responds as follows:

```
    Dump Cycle List Operation? [list] ↩
```

We hit New Line to select the default, list. Possible responses to this query are:

list	Print the list.
forward	Move the current position forward.
backward	Move the current position backward.
reset	Move the current position to the top of the list.
end	Stop editing the list.

We provide a default dump cycle list. The entries are:

cycle Lists the cycle letters that correspond to those in **/etc/fstab**, which indicate when the file systems will be dumped. For instance, file system **/comm** might be set to **w**, so it is only dumped once per week. You set the cycle letter in **/etc/fstab** when you add the file system with **addfsys** in Procedure 8.1.

level These numbers are used internally by the **fsdump** program. The ones we supply need not be changed for normal system operation.

Multi Multi-dumps. **fsdump** normally dumps one file system per tape. If you specify "y", then multi-dumping occurs. This means as many file system as there is room for will be written to tape. An "n" entry means write just one file system per tape, as is shown for the monthly line below.

label We recommend that you label your tapes so that they match the entries in the dump cycle list. `Monthly` means dump all file systems marked with **d**, **w**, or **m** (daily, weekly, or monthly). `Monday Set` means dump all file systems marked with **d** (daily), and so on with the other weekdays. Friday's dump becomes part of the `Weekly Set` of dump tapes.

When you select the default **list** entry, the system responds with this list as follows:

```
    Dump Cycle List

    Cycle   Level  Multi        Label
    -----   -----  -----  --------------------
    dwm       0      n     Monthly
--> d         6      y     Week 1 - Monday Set
    d         7      y     Week 1 - Tuesday Set
    d         8      y     Week 1 - Wednesday Set
    d         9      y     Week 1 - Thursday Set
    dw        1      y     Week 1 - Weekly Set
    d         6      y     Week 2 - Monday Set
    d         7      y     Week 2 - Tuesday Set
    d         8      y     Week 2 - Wednesday Set
    d         9      y     Week 2 - Thursday Set
    dw        2      y     Week 2 - Weekly Set
    d         6      y     Week 3 - Monday Set
    .         .      y       .            .
    .         .      y       .            .
    .         .      y       .            .
    dw        5      y     Week 5 - Weekly Set


    The current position is marked by -->.
```

To the **forward** response, the system responds:

```
How far to move forward? [1] ↵
```

Take the default or enter the number of positions you wish to move forward. This number must be greater than 0. If your number moves the pointer off the list, the pointer is positioned at the top of the list again.

```
    Dump Cycle List

    Cycle   Level  Multi        Label
    -----   -----  -----  --------------------
    dwm       0      n      Monthly
    d         6      y      Week 1 - Monday Set
--> d         7      y      Week 1 - Tuesday Set
    d         8      y      Week 1 - Wednesday Set
    d         9      y      Week 1 - Thursday Set
    dw        1      y      Week 1 - Weekly Set
    d         6      y      Week 2 - Monday Set
    d         7      y      Week 2 - Tuesday Set
    d         8      y      Week 2 - Wednesday Set
    d         9      y      Week 2 - Thursday Set
    dw        2      y      Week 2 - Weekly Set
    d         6      y      Week 3 - Monday Set
    .         .      y        .            .
    .         .      y        .            .
    .         .      y        .            .
    dw        5      y      Week 5 - Weekly Set


    The current position is marked by -->.

    Press the New Line key to see the fsmgmt menu [?, ^, q]:
```

The **reset** choice displays the top entry in the dump cycle list.

Procedure 8.8: Make File System Backup Tapes

Purpose	To dump file systems to tape as indicated by the Dump Cycle List in Procedure 8.7: Modify Dump Cycle.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**fsdump**
References	**dump2**(1), **dumptab**(4), **dump2label**(1)

Use the **fsdump** command when you want to create dump tapes of file systems. The **fsdump** command uses the dump cycle list and determines the correct entry for the current day. Then, it looks through the file system table (**/etc/fstab**) and locates those active file systems that match the current cycle entry; when a match is found, the file system is scheduled to be dumped. The **fsdump** command calls the **dump2**(1) program for each file system that is scheduled to be dumped.

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

```
Tape Drive? [0] ↵

Tape Medium? [qic] ↵
```

Notice that we took all the defaults above. The default tape medium represents a QIC-150 4" cartridge tape.

Notice that only the multi-dump field monthly line is set to "n". This is to ensure that the monthly dumps are separated in case of loss or damage to the tape. Remember, the "n" setting means that **fsdump** will list each file system that is scheduled to be dumped, one at a time, and give you the option of dumping it or not dumping it (skip). When the multi-dump field is set to "y", then all file systems would be listed.

Assume that **/usr/opt** is the first file system to be dumped and the dump cycle pointer is on `Monthly`. You can label your tape sets as prompted:

```
Label the next tape set:

Monthly Set
/usr/opt
06/01/89
Tape #1, #2, ...
```

```
        Mount the first tape of the set and place the tape drive online.
        Enter yes when the tape is ready.
        Is the tape ready? yes ↵
```

Possible responses to the last query are **yes, no,** or **skip**. You should skip a file system only if there is an error which makes it impossible to back it up or if the system crashed while **fsdump** was running. For instance, if you are dumping five file systems and the system crashes while you're on the third one, you would skip numbers one and two and resume dumping with the third file system.

When you answer yes, the dump will begin:

```
        Dumping: /usr/opt  Monthly Set  06/01/89
```

When you are finished dumping, the pointer on the dump cycle list advances.

Procedure 8.9: Restore File Systems from fsdump Tapes

Purpose	To restore file systems to disk from backup tapes that were made with the **dump2**(1) command.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**fsrestore**
Note	The directory in which the restore is to take place must be empty.
References	**dump2**(1), **restore**(1)

Use this command when you want to load files or file systems from tape to a directory on disk. You might want to do this when you're recovering from a disk failure or moving an entire file system from one disk to another. You are asked to supply the directory name where the restore will be done. A full restore works by restoring a full (monthly) dump onto an empty file system. Then, incremental (weekly and daily) dumps are restored on top of that until the most recent tape set has been loaded. You should locate the following tapes associated with the desired file system:

1) The most recent monthly dump.
2) All weekly dumps since the most recent monthly dump.
3) All daily dumps since the most recent weekly dump.

When **fsrestore** asks you for tape sets, you should restore them in the order shown here (a monthly followed by weeklies followed by dailies). You will be asked to mount the first tape of a set and then execute the restore program which will ask for further tapes from the set. This process will be repeated until you indicate that there are no more tapes.

Mount the tape and select **fsrestore**; the system responds as follows:

```
Tape Drive? [0] ⤶
```

Next, you are asked for the name of the *empty* directory in which you will restore the file system.

```
Mount Directory Name? /foobar ⤶
```

```
        Is the first tape of the set ready? [y] ↵
```

A **yes** answer means that you have mounted the tape and the drive is online. The restore process begins.

If **/usr/newdir** is not mounted, you will be instructed to use **mountfsys** to mount it. If **/usr/newdir** is not empty, you will be instructed to either choose another directory or to use **makefsys** to clear the file system.

Next, the available file systems on the tape are displayed, then you are queried about the file system you typed in:

```
     The file systems on this tape are:

     /foo
     /foobar
     /usr/newdir

     File Systems from Tape? [/foobar] ↵
```

If there are problems, an error message will be printed, otherwise you will be asked for another tape:

```
        Is there another tape set? [yes] ↵
```

Pressing the New Line key enters the **yes** default and the restore process continues. When you have loaded all tape sets, answer **no** when you are asked for another tape set, and you will exit from **fsrestore**.

There will probably be times when there is only one file system on the tape and you want to write that file system to another location other than its current pathname. For instance, you might want to put **/foo** (which is on the tape) into **/new_location/newfoo**. The system would respond as follows:

```
     Mount Directory? /new_location/newfoo ↵

     The file system on the tape /foo does not match the
     mount directory /new_location/newfoo.

     Restore this tape set? [n] ↵
```

This writes the file system on the tape to the new location.

Procedure 8.10: Extract Individual Files from fsdump Tapes

Purpose	To restore individual files that have been lost or deleted.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**filerestore**
Note	This command restores files that were previously dumped with **fsdump**.
References	**restore(1)**

Use this command for small restore tasks, such as when a user has accidentally deleted files or directories. For major file system recovery, such as after a disk failure, use **sysadm fsrestore**.

The **filerestore** command asks for the directory where the restored files should be placed. If this directory is invalid, it asks if you want the directory to be created. If you don't create the directory, you will be asked to select another directory.

In the following, let's assume that user Smith has accidentally deleted a file named **redeye** in directory **/sales/accounts/smith**. We'll restore that file in **/tmp**.

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

```
Tape Drive? [0] ↵

Restore directory? /tmp ↵

sysadm will now run the restore program in interactive mode
on repeated sets of dump tapes until you tell it to stop.
If you haven't run filerestore before, enter ? when the
'restore>' prompt appears.

Please mount the first tape set.
Is the first tape of the set ready?  y ↵

The file systems on this tape are:

/sales/specs
/sales/accounts
```

```
        File system from Tape?  /sales/accounts ↵
```

You can use the following commands at the *restore>* prompt:

ls	list directory
cd	change directory
pwd	print working directory
add	add filename to the list of files to be extracted
delete	delete filename from the list of files to be extracted
extract	extract requested files
quit	exit program
help	print this list

We'll use some of the commands from the list:

```
        restore> cd /sales/accounts/smith ↵

        restore> ls -al redeye

                -rw-rw-rw-  1  smith  doc  821 Jun 10 14:29 redeye

        restore> add redeye

                redeye has been added.

        restore> extract

        You have not read any tapes yet.  Unless you know which
        volume your files are on, you should start with the last
        volume and work towards the first.

        Specify next volume #: 1 ↵
        Set owner/mode for '.'? [yn] n ↵

        restore> quit ↵
```

Above, we changed directory to the working directory, listed the file, added the file to the list to be extracted, and extracted the file. It was written to **/tmp** so that you can inspect it before installing it in its original directory.

Procedure 8.11: Add a Swap Entry to /etc/fstab

Purpose	To add swap entries to the **/etc/fstab** file.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**addswap**
References	**swapon(1M)**

A swap entry is a line in **/etc/fstab** of the form:

```
/dev/dsk/swap   swap   swap   sw  x  0
```

The name of the logical disk is user-defined (the first field). The other fields are set by **sysadm**. Swap entries are used by **swapon**(1M) and by the accounting system. These apply only to local logical disks or NFS-mounted swap files.

Below, we'll add **comm** with **addswap**. We begin by typing:

```
# sysadm addswap ↵
```

The system responds as follows:

```
Logical Disk? comm ↵

Swap entry for comm has been added.
```

Procedure 8.12: Delete a Swap Entry from /etc/fstab

Purpose	To delete swap entries from the **/etc/fstab** file.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**delswap**
References	**swapon(1M)**

This function deletes a swap area from **/etc/fstab**. If the file system has been enabled for swapping by the **swapon** command, it will be used until the system is shut down. The DG/UX system does not permit a swap area to be removed from activity. Below, we'll delete **comm**. We begin by typing:

```
# sysadm delswap ↲
```

The system responds as follows:

```
Logical Disk?  comm ↲

Swap entry for comm has been deleted.
```

Procedure 8.13: Making Tapes

Purpose	To move online data to a magnetic tape.
Starting Conditions	administrative or multiuser state Diskless machines must specify a remote tape drive with this command.
sysadm menu	fileinfo
Commands	**cpio(1), find(1)**

This procedure is done without **sysadm**. When you have a small-scale dumping task, such as making a personal tape for a user, you don't need to use the **dump** and **restore** operations. You can use **cpio(1)** instead. See the **cpio** reference page for a complete listing of options and further instructions. To dump a directory named **/sales/smitht** (and all subdirectories and files), do the following:

1) Mount a tape and put the drive online.
2) Go to the directory you wish to dump:

```
# cd /sales/smitht ↵
```

3) Dump everything in the directory to tape:

```
# find . -print | cpio -ocvB  >  /dev/rmt/0 ↵
```

The contents of **/sales/smitht** have been dumped to tape. To write individual files to tape, go to the directory where the files are located and type

```
# ls fileA fileR fileZ | cpio -ocvB> /dev/rmt/0 ↵
```

The contents of all three files are dumped to tape with this command. We used the following options to **cpio** in the above command lines:

o	Copy files to standard output
c	Use ASCII headers for portability
v	Be verbose: prints a list of filenames
B	Use large type blocks (5120 vs. 512 bytes)

We directed the output of the dump to raw magnetic tape (rmt), device 0. of 0.

Expert File System Information

Information in this expert section is optional. You do not need to read this section to manage your file systems.

Mounting and Unmounting a File System Without sysadm

As we stated earlier, to access the file system on logical disk **thor**, it must be *mounted* on a directory. You can mount a file system from the command line with the **mount(1M)** command. The following command line shows the file system on logical disk **thor** being mounted on a directory called **/usr/thor**. Notice that the **mount** command identifies the logical disk by its node name.

```
# cd /usr ↵
# mkdir thor ↵
# mount /dev/dsk/thor   /usr/thor ↵
```

Above, we created mount directory **/thor** before we mounted the file system on it. You can do the same thing with the **mountfsys** command in this chapter. To unmount a file system, you can use the **umount**(1M) command.

Dumping and Restoring Without sysadm

Refer to the **dump2**(1) and **restore**(1) manual pages for detailed information.

Using dump2(1)

The **dump2** program copies some or all files on a logical disk to the backup medium based on the dump "level". There are 10 levels: 0 through 9. Execute the **dump** program by specifying a logical disk and a dump level as in the following:

```
# /etc/dump2  0ufd  /dev/rmt/0 6250 /dev/dsk/root
```

The above command line uses the following options:

0	dump level
u	update the **/etc/dumpdates** file
f	dump filename
d	tape density

The **dump2(1)** manual page lists all available options.

The dump level number instructs **dump** to make a copy of each file that has been modified since the most recent dump at any lower dump level number. For example, if the dump level is 3, **dump2** will make a copy of any file that has been modified since the most recent level 0, 1, or 2 dumps. Level 0 dumps *every* file in the file system because there is no lower dump level. A level 0 dump is often called a "full" dump.

The**dump2** command knows that a file has been modified by examining the inode change time (or "ctime") and the file modification time (or "mtime") for each file (see **stat(2)** for detailed information). If either of these is later than the dump time for the file system at the appropriate dump levels, then the file has been "modified" since the previous lower level dumps. The **dump2** command knows when the file system was last dumped at any given level because it keeps this information in the file **/etc/dumpdates**. This file contains lines of the form:

```
/dev/dsk/root        3 Fri Jan  6 15:45:15 1989
```

In this case, the most recent level 3 dump for **/dev/dsk/root** was made at 3:45 p.m. on 1/6/89. An entry is added to **/etc/dumpdates** only after the dump completes the successfully. This prevents it from inserting a date for a dump that later aborts. Also, duplicate entries are deleted. In the example above, any other level 3 entries for **/dev/dsk/root** would be deleted when adding the new one.

To make backup tapes, you must first set up a dump schedule and determine which dump level should be used on each dump to implement that schedule. For example, the default dump cycle list shipped with **sysadm** is based on the following dump schedule:

- A "monthly" dump is performed once a month (probably on the weekend). This records *every* file on the file system.
- Each Friday a weekly dump is performed. This records every file modified since the most recent weekly or monthly dump.
- On Monday, Tuesday, Wednesday, and Thursday, a daily dump is performed. This records every file modified since the most recent daily, weekly, or monthly dump.

These dumps would generally be done after prime working hours (say at 6 p.m.).

Dump levels that would implement this schedule are

DUMP	**LEVEL**
Monthly	0
Weekly 1	1
Weekly 2	2
Weekly 3	3
Weekly 4	4
Weekly 5	5
Monday	6
Tuesday	7
Wednesday	8
Thursday	9

The first weekly dump is the one following the monthly dump. This schedule would be carried out (say, for the root file system) by performing dumps in the following order:

DUMP	**COMMANDS**
Monthly	**/etc/dump2 0ufd /dev/rmt/0 /dev/dsk/root**
Monday (1)	**/etc/dump2 6ufd /dev/rmt/0 /dev/dsk/root**
Tuesday (1)	**/etc/dump2 7ufd /dev/rmt/0 dev/dsk/root**
Wednesday (1)	**/etc/dump2 8ufd /dev/rmt/0 /dev/dsk/root**
Thursday (1)	**/etc/dump2 9ufd /dev/rmt/0 /dev/dsk/root**
Weekly (1)	**/etc/dump2 1ufd /dev/rmt/0 /dev/dsk/root**
Monday (2)	**/etc/dump2 6ufd /dev/rmt/0 /dev/dsk/root**
Tuesday (2)	**/etc/dump2 7ufd /dev/rmt/0 /dev/dsk/root**
Wednesday (2)	**/etc/dump2 8ufd /dev/rmt/0 /dev/dsk/root**
Thursday (2)	**/etc/dump2 9ufd /dev/rmt/0 /dev/dsk/root**
Weekly (2)	**/etc/dump2 2ufd /dev/rmt/0 /dev/dsk/root**

and so on.

Week 5 will not always be needed, depending on the month.

Changing the Dump Cycle List

You should read the previous section to understand how **dump2**(1) works. The dump cycle list is kept in the file **/usr/admin/tables/dump_cycle**. It contains lines of the form:

```
cycle_pattern       dump_level          tape_label
```

where the fields are separated by tabs. The first field is a **grep(1)** pattern that selects one or more letters from the dump cycle field in **/etc/fstab**. The permissible letters are:

m	monthly
w	weekly
d	daily
x	ignore

A single letter or a pattern may be used:

d	match d entries
dwm	match d, w, or m entries

The second field is a dump level number, which is a digit between 0 and 9. The last field is an arbitrary text string which is given to the operator as the desired tape label for this dump.

One line in the **dump_cycle** file is marked as the "current" dump by placing an at-sign in front of it:

```
@[dwm]     0          Monthly Set
```

When **sysadm fsdump** is used, it locates the current entry. It then locates every line in **/etc/fstab** whose dump cycle field matches `cycle_pattern`. In this example, the pattern `[dwm]` would match d, w, and m entries. It then executes the **dump(1)** command once for each matching line in **/etc/fstab**. The dump level is taken from the second field in the dump cycle and the tape label given to the operator is taken from the third field.

The **dump_cycle** file (and thus **fsdump**) can be set up to use any dump schedule. For example, if a site wishes to do a full dump every Monday and do incremental daily dumps on Tuesday through Friday, the following **dump_cycle** file would suffice:

```
@[dwm]     0  n     Monday (Full) Set
d          1  n     Tuesday Set
```

```
d         2  n      Wednesday Set
d         3  n      Thursday Set
d         4  n      Friday Set
```

Using restore(1)

Restore file systems using the **restore**(1) command with the **-r** option. If a file system is completely destroyed, it can be restored by first remaking the file system and then using the **restore** command on the following tape sets:

1) The most recent monthly dump.

2) All weekly dumps made since the most recent monthly dump.

3) All daily dumps made since the most recent weekly dump.

For example, if the file system is lost on Wednesday of the second week (before the Wednesday dump), the following tapes are needed:

Monthly
Weekly (1)
Weekly (2)
Monday (2)
Tuesday (2)

The following are sample command sequences:

Remake and check the file system:

```
    # /etc/unmount /dev/dsk/foo ↵
    # /etc/mkfs /dev/dsk/foo ↵
    # /etc/fsck /dev/dsk/foo ↵
```

Mount and restore the monthly tape set:

```
    # /etc/mount /dev/dsk/foo /mount_name ↵
    # cd /mount_name ↵
    # /etc/restore -rf /dev/rmt/0 ↵
```

Restore the weeklies, dailies, etc.:

```
    # /etc/restore -rf /dev/rmt/0 ↵
```

The **restore -r** command restores all files in the current directory.

End of Chapter

Chapter 9
File Information

This chapter shows you how to find and display information about files in your file systems.

File Terms

We use the following terms in this chapter:

inode	Data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID. The number of inodes represents the total number of files that can exist on the system. The total number of inodes is set in the **diskman** program. An inode is 126 bytes long, and there are 4 inodes to a disk block.
disk block	A unit of data as it is actually stored and manipulated. It consists of 512 bytes.
setuid	A mode bit that can be specified for any executable file. When a user runs an executable file that has the setuid bit set, the system gives the user the permissions of the owner of the executable file for the duration of the command. See **chmod(1)**.
/dev	Administrative directory containing entries for all devices on the system.

File Information Procedures

When you select **fileinfo** from the **sysadm** Main Menu, the following choices are displayed:

```
                         File Information

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

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

Remember: HELP is always available for any of the commands in a **sysadm** menu. For example, you can get help on **diskuse** by typing **1?**. Also, type ? for any query you don't understand how to answer.

Procedure 9.1: List Disk Information

Purpose	To obtain usage information on mounted file systems.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**diskuse**
References	**df(1M), du(1M)**

Use this command to print a table showing the number of blocks and inodes in use on each mounted file system. If you select **diskuse**, the system responds as follows:

```
                 Free      Total     Pct   Free     Total    Pct
File System      Inodes    Inodes    Used  Blocks   Blocks   Used
---------------  --------  -------   ----  -------  -------  ----
/                 3738      4094       8%   2204     18480   88%
/usr             19534     22526      14%  28012    152240   81%
/udd/sys5        17772     22526      21%  33100    152240   78%
/comm                0         0       0%  116926   584110   79%
/usr/local        1475      7650      98%   9614    88000    89%
/spare           20474     20474       0%  135272   140800    3%
/acme             46231     7500      38%   27433   58700    53%
```

Procedure 9.2: Search Files by Age

Purpose	To list all files in a directory by specified age.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**fileage**
References	**ls(1)**

You will find it necessary to search your directories from time to time to see what files are taking up space, but receiving little use. For instance, you might want to remove a file that has not been accessed in the last 90 days. This command searches a specified directory and lists all files that have not been accessed within a given number of days. When you select **fileage**, the system responds as follows:

```
Search Directory?  /comm/smitht ↵
```

You must enter a pathname starting with /. The entire tree starting with / will be searched. There is no default directory. If there is a problem with your entry, you will get one of the following messages:

```
/comm/smitht does not exist.
```

or

```
/comm/smitht is not a directory.
```

Next, you are queried:

```
File age (in days)? [90]  ↵
```

You can enter any number of days or choose the default. All files that have not been accessed in the period you specify will be printed.

```
Owner      Size (bytes)     Last Access        File Name
-------    -----------      -----------     -----------------
smitht       150227         May 14 1989     /comm/smitht/a.out*
smitht          400         Nov 03 1987     /comm/smitht/emacs
smitht         2000         Aug 21 1986     /comm/smitht/nfs.1
smitht       765555         Jun 11 1986     /comm/smitht/tcpip
smitht      9999999         Dec 29 1985     /comm/smitht/xwindow
```

Procedure 9.3: List Files by Name

Purpose	To locate and list specific files in a directory.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filename**
References	**ls(1), grep(1)**

This command searches a directory and lists all files which have a given basename. That is, if the desired name is "foo", **filename** will print the pathnames of files that match it, such as **/usr/foo** and **/usr/lib/foo**. This is useful when you need to locate a file in a directory tree. You can specify the wildcard search and locate all files beginning with the same string. Below, we are looking for all filenames beginning with `lost.file` so we attach a * to the filename.

When you select the **filename** command, the system responds as follows:

```
Search Directory?  /comm ↵
File Name?  lost.file* ↵

Owner     Size (bytes)  Last Access      File Name
----      -----------   -----------   ----------------------
smitht    155           Apr  2 1988   /comm/docs/lost.file1
smitht    3410          May  9 1989   /comm/docs/lost.file2
smitht    1550          May 10 1989   /comm/docs/lost.file3

Press the New Line key to see the fileinfo menu [?, ^, q]:
```

If you specify a file name that is not in a given directory, a message is printed telling you the file was not found.

Procedure 9.4: List Files with Permission Errors

Purpose	To search for and list files in a directory that are owned by root and have the setuid bit set. To search for and list device files that should be in **/dev**.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filescan**
References	**setuid**(2), **chmod(1)**

This command searches a directory tree for files that have suspicious ownership and permission settings. These files may indicate that an error or a security breach has occurred. This command searches the directory you specify and reports files that have the following problems:

- Device files that exist outside of **/dev** --

  There should be no device files outside **/dev** unless you, as system administrator, have created or moved device files for a special purpose, such as a test.

- Files that are owned by root and have the setuid bit set --

  Only root can set a file to setuid. The purpose of the setuid bit is to allow a program to have temporary access to a more privileged status (such as root) while it is running. This is a very useful part of the DG/UX system because it allows controlled access to many facilities that you would not want users to get at directly.

If a command file, such as **/bin/sh** has the setuid bit set, then anyone executing that command will have superuser permissions for the duration of the command's execution. For instance, if user **tom** obtains a copy of **sh**, then *tom* is given the owner permissions of root (superuser). Look at the sample listings below and note the "suspicious" file owned by root in **/usr/tom**:

```
-r-sr-xr-x  1 root  bin     38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x  1 root  bin     19812 Aug 10 16:05 /usr/bin/crontab
-r-sr-xr-x  1 root  sys     11416 Aug 11 01:26 /bin/mkdir
---s--x---  1 root  users   45376 Aug 18 11:08 /usr/tom/bin/sh
```

Above, all of these files have the setuid mode set as indicated by the `s` in each permissions row. We call the last entry suspicious because no user should have a personal copy of **/bin/sh**. With this copy, user **tom** can execute **/usr/tom/bin/sh** and become superuser.

To protect your system, never leave your logged-in terminal unattended. Another user could move or copy files, or do whatever he or she wanted with *your* superuser privileges.

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

```
Search Directory? /crock ⤶

The following files look suspicious.
The problem codes are:

device  This device file was found outside of /dev.
setuid  This file is owned by root and has setuid mode on.


Problem         File Name
-------         --------------
setuid          /crock/sh
setuid          /crock/ps
device          /crock/tty14
```

When you find a setuid bit set, investigate further. You may need to correct setuid permissions with the **chmod(1)** command. In general, you will not be creating device files, so there shouldn't be any ouside of **/dev**. There might, however, be the case when you or someone else creates a *test* device file outside of **/dev**. If necessary, you can either move or delete the device file.

Procedure 9.5: List Files by Size

Purpose	To search for and list files in a directory by specified size.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filesize**
References	**df(1M), ls(1)**

Use this command mainly for cleanup purposes. If, for instance, you notice a drastic reduction in available disk space, you could check to see what large files might be using up the space. In a specified directory, the **filesize** command lists the largest files. You supply the directory name and the number of files. If the directory exists, the matching file names are printed. The default number of files is 10. This means that the 10 largest files will be displayed.

When you select the **filesize** command, the system responds as follows:

```
Search Directory? /comm/smitht ↵

How many files? [10] 3 ↵
```

We entered 3 to this query. The system displays the 3 largest files.

```
Owner        Size          Last Access          File Name
-------    ---------      -----------       -----------------
smitht      456000        Dec 29 1987       /comm/smitht/tokyo
smitht      200998        Oct 10 1988       /comm/smitht/new_york
smitht      100001        Sep 21 1988       /comm/smitht/mayberry

Press the New Line key to see the fileinfo menu [?, ^, q]:
```

End of Chapter

Chapter 10
tty Management

This chapter shows you how to set up and manage terminals (ttys) on the DG/UX system with **sysadm ttymgmt**. If you are not already familiar with **inittab**(4) and **gettydefs**(4), you may want to consult these manual pages as you use the information in this chapter.

The major sections of this chapter are:

- TTY Terms
- TTY Management Procedures
- Expert TTY Information

TTY Terms

We use the following terms in this chapter:

tty	Derived from the abbreviation for teletypewriter, the term covers the whole area of access between the DG/UX system and peripheral devices, including the system console. A tty is commonly known as a terminal. It shows up in commands such as **getty**(1M) and **stty**(1), in the names of device special files such as **/dev/tty01**, and in the names of files such as **/etc/gettydefs**, which is used by **getty**.
tty line	The physical equipment through which access to the computer is made. Also known as a terminal line or port.
linesets	A group of settings (modes) in **/etc/gettydefs** that govern baud rate, special characters, echoing, etc., as defined in **termio**(7). The tty line and the terminal must be in the same mode before communication can take place. Lineset names usually refer to baud rate, such as 9600.
baud rate	The speed at which data is transmitted. Baud refers to the number of bits transmitted each second over a line.

hunt sequence	A lineset structure that allows a user to try more than one lineset until a match is found between a tty line and a terminal. Used mainly for dial-up lines. A default sequence is shipped with the DG/UX system.
login state	On = login. Off = no login. The on state is for login ttys. Use the off state if you want to disable the line, or if you want to use the line for a printer.

TTY Management Procedures

If you select **ttymgmt** from the Main Menu, the system displays the following:

```
                TTY Management

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

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

Procedure 10.1: Define TTY Default Settings

Purpose	To set terminal defaults.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**ttydefaults**
References	**tty**(7), **inittab**(4)

This function defines the default values for other **ttymgmt** functions. These include login state, lineset name, hangup delay, TERM variable, and a description (optional) for every terminal. These defaults come with values already set. You should verify that each of these defaults is correct for your system; if so, hit the New Line key. If not, set the value you want. When you select **ttydefaults**, the system responds as follows:

```
Default Login State? [on] ↵

Default Lineset Name? [9600]  ↵

Default Hangup Delay (in seconds)? [0] ↵

Default TERM Variable?  [vt100]  ↵

Default Description? ↵

The tty defaults have been set.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Procedure 10.2: Add a Single TTY

Purpose	To add one terminal at a time to the system.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**addtty**
References	**tty**(7), **inittab**(4)

The tty name must refer to an existing device file in the **/dev** directory. The terminal controllers, such as a Systech Asynchronous Controller (syac), on your CPU will determine how many tty entries are in **/dev**. These device files are created automatically by the kernel at boot time. Files are of the form **/dev/ttyN** where N is a number consisting of two or more digits, such as in tty01, tty02, tty03, tty100, etc.

So, when you make a tty available for login use, you are putting a tty name that already exists in **/dev** into the tty table in **/etc/inittab**.

To add a tty to your system, select the **addtty** subcommand. The system responds as follows:

```
TTY Number?  ? ↵
```

We typed ? because we were unsure what ttys were available. The system responds by listing them as follows:

```
tty00 through tty15

TTY Number? 01 ↵

Login state? [on]  ↵
```

By pressing New Line, we got the default *on* state for logins. To use this tty for some other purpose (such as a laser printer) answer *off*.

```
Lineset Name? [9600] ? ↵

The lineset defines the terminal modes and login banner used
for logging in.  The available linesets are:

300  1200  2400  9600

Lineset Name? [9600] ↵
```

```
Hangup Delay (in seconds)? [0] ↵
```

The hangup delay specifies how many seconds the **getty(1M)** program will wait for the user to enter the login name. Delays are often used on dial-up lines to avoid wasting telephone connect time; the delay is generally 60 to 120 seconds. A delay of 0 is often used on direct tty lines. The maximum delay is 600 seconds (ten minutes).

```
    TERM Variable? [vt100] ↵

    Available in Init Administrative State? [no] ↵

    Description? ↵

    TTY 01 has been added.

    Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

We selected the default TERM value. You may define the initial value of the TERM variable for this tty. This environment variable is used by programs like **vi** to determine the type of terminal that is on this tty. The value of TERM will be exported to whatever login program is used by the person logging on to tty01. If you add a TERM value that does not correspond to an entry in **/usr/lib/terminfo**, you will receive an error message.

Procedure 10.3: Delete TTYs

Purpose	To delete a tty entry from the **inittab** table.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**deltty**
References	**inittab**(4)

You might want to delete a tty if you have removed a terminal controller board, thus reducing your tty capacity. But generally, you will not delete tty entries. To do so removes them entirely from **etc/inittab**. If you want to disable a tty, use the **modtty** command and turn the login state to *off*.

To delete a tty, select the **deltty** subcommand. The system responds as follows:

```
TTY Number? 12

Do you really want to delete tty 12? [no] y

TTY 12 has been deleted.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Procedure 10.4: Modify TTYs

Purpose	To modify an entry in the **inittab** table. To set an administrative state CRT.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**modtty**
References	**inittab**(4)

Use this function to change settings on ttys. In particular, you can use this function to make a CRT available when the system is in run level 1, the administrative state.

The tty to be modified must already have an entry in **/etc/inittab**. The default for each query in this function is whatever value you have set previously.

To modify a tty, select the **modtty** subcommand. The system responds as follows:

```
TTY Number?  14 ↵

Login state? [on]  off ↵

Lineset Name? [9600] ↵

Hangup delay (in seconds)? [60] ↵

TERM variable? [vt100] ↵

Available in Init Administrative State? [no] ↵

Description? Line disabled 4/21/89 ↵

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

We chose the "off" login state, the default lineset, 60 seconds for the hangup delay, and the default TERM variable. The `Description` query is there for your benefit; you can leave it blank by pressing New Line or you can enter a description as we did above.

Procedure 10.5: List TTYs

Purpose	To list tty entries in the **inittab** table. To give tty information.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**lstty**
References	**inittab**(4)

To list ttys and get information about them, select the **lstty** subcommand. The system responds as follows:

```
TTY Numbers? [all] ↵

The list of available ttys is:

                        Hangup      Line
  TTY       State       Delay       Set         Description
 ------     ----        -----      -------      -----------
  00         on           0          9600       Admin Terminal
  01         on           0          9600       C. Roach
  02         off          0          9600       C. Robin

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Procedure 10.6: Add Multiple TTYs

Purpose	To add more than one terminal at a time to the system.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**installtty**
References	**tty**(7), **inittab**(4)

A tty name must refer to an existing device file in the **/dev** directory. The terminal controllers on your CPU will determine how many tty entries are in **/dev**. For instance, if you have 32 lines, then you will have 32 tty device files in **/dev**. These device files are created automatically by the kernel at boot time. Files are of the form **/dev/ttyN** where N is a number consisting of two or more digits, such as in tty00, tty01, tty02, tty100, etc.

So, when you make a tty available for login use, you are putting a tty name that already exists in **/dev** into the tty table in **/etc/inittab**.

The procedure for adding tty lines depends on the number of tty entries that you have in **/dev**. If you have less than 64, then use this procedure as described below. If you have more than 64, then you need to run **sysadm newdgux** and change the NPROC variable to suit your needs. NPROC determines the maximum number of processes that can be active at one time on your system. If, for example, you have 84 ttys, then you need to adjust the NPROC variable upward by 20 from its current default value. This will prevent the process table from overflowing when processes are started on the ttys. After running **newdgux**, reboot your system to initialize the new kernel, then run **sysadm installtty** which spawns a **getty** process on every available tty line (all tty entries in **/dev**). If you have **getty** processes running on unused lines, you can edit **/etc/inittab** and change the "respawn" field to "off" for those you don't want activated.

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

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

Login State? [on] ⮐
Lineset Name? [9600] ⮐
```

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

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

Ready to install ttys? [yes] ↵
Sysadm will now create the tty entries...
Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Expert TTY Information

The remaining sections contain expert information describing alternative methods for tty administration. You do not need to read these sections to manage your system; we provide them for readers who want more details.

How the TTY System Works

A series of four processes connects a user to the DG/UX system: **init**(1M), **getty**(1M), **login**(1) and **sh**(1) or **csh**(1). The **init** program is a general process spawner that is invoked in the boot procedure. It spawns a **getty** process for each line that a user may log in on, guided by instructions it reads from **/etc/inittab**. The **getty** program requires the name of a special file from **/dev**, such as **tty01**, as a *line* argument.

A user attempting to make a connection generates a request-to-send signal that is routed by the hardware to the **getty** process for one of the tty line files in **/dev**. The **getty** process responds by sending an entry from file **/etc/gettydefs** down the line. The **gettydefs** entry used depends on the *speed* argument used with the **getty** command. (In the SYNTAX of the **getty(1M)** command, the argument name is *speed*, but it is really a pointer to the *label* field of a **gettydefs** entry.) If there is no *speed* argument, **getty** uses the first entry in **/etc/gettydefs**. The login prompt is among the fields in the **gettydefs** entry.

On receiving the login prompt, the user enters a login name. Then **getty** starts a program named **login**, using the login name as an argument. The **login** program sends the prompt for a password, evaluates the user's response against the **passwd** file, and assuming the password is acceptable, calls in the user's shell as listed in the **/etc/passwd** entry for the login name. If no shell is named, **/bin/sh** is furnished by default. Upon login, a user's shell executes **/etc/profile** for **sh** users, and executes **/etc/login.csh** for **csh** users.

The **/bin/sh** program executes the user's **.profile** for **sh** and **/bin/csh** executes the user's **.login** for **csh**. These files often contain **stty** commands that allow a user to reset terminal options should they not want the defaults.

Modifying TTY Linesets in /etc/inittab

You have two ways to modify tty linesets:

- Use the **sysadm modtty** command. This command leads you through a series of prompts. Your responses edit a "getty" entry in **/etc/inittab**.

- Use a text editor to edit **/etc/inittab**.

The **/etc/inittab** file contains instructions for the **/etc/init**(1M) command. The general format of a line entry in the **/etc/inittab** file is as follows.

identification:level:action:process

The four colon-separated fields are as follows:

identification	A unique one- or two-character identifier for the line entry.
level	The run level in which the entry is to be performed.
action	How **/etc/init** treats the process field.
process	The shell command to be executed.

/etc/inittab contains several entries that spawn **getty** processes. The following example is a selection of such entries from an **/etc/inittab** file.

```
con::respawn:/etc/getty console console
01:23:respawn:/etc/getty tty01 vt100 9600
02:23:respawn:/etc/getty tty02 vt100 9600
03:23:respawn:/etc/getty tty03 vt100 9600
04:23:off:/etc/getty tty04 vt100 9600#line not in use
05:23:off:/etc/getty tty05 vt100 9600#line not in use
```

There are at least three things you might want to do to an **inittab** entry for a tty line:

- Change the action. Two actions that apply to tty lines are "respawn" and "off" (see the **inittab**(4) manual page for complete information on this field).
- Add or change arguments to **/etc/getty** in the process field. A frequently used argument is **−t***nn*. This tells **getty** to hang up if nothing is received within *nn* seconds. It's good practice to use the **−t** argument on dial-up lines.
- Add or change comments. Insert comments after a pound sign (**#**).

Setting Terminal Options

Once the user has successfully logged in, he or she may find that certain terminal options would be preferable to ones in the default set.

The command to control terminal options is **stty**(1). Many users add an **stty** command to their **.profile** (or **.login** for **csh**) so the options they want are automatically set as part of the **login** process. Here is an example of a simple **stty** command.

```
$ stty cr0 nl0 echoe −tabs erase ^H
```

The options in the example mean the following:

cr0 nl0	No delay for carriage return or new line. Delays are not used on a video display terminal, but are necessary on some hardcopy terminals to allow time for the mechanical parts of the equipment to move.
echoe	Erase characters as you backspace.
−tabs	Expand tabs to spaces when printing.
erase ^H	Change the character-delete character to a ^H. The default character-delete character is the pound sign (#). Most terminals transmit ^H when the **backspace** key is pressed.

Interpreting Linesets in /etc/gettydefs

The **/etc/gettydefs** file contains information used by the **getty**(1M) command to establish the speed and terminal settings for a line. The general format of the **gettydefs** file is

```
label# initial-flags # final-flags #login-prompt #next-label
```

The following are sample lines from a **gettydefs** file:

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600

9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800

4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400

2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200

1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300

300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
```

The entries above form a single, circular hunt sequence; the last field on each line is the label of the next line. The `next-label` field for the last line shown points back to the first line in the sequence. The object of the hunt sequence is to link a range of line speeds so that when a user tries to log in on a dial-up line, say at 300 baud, the system will be able to match that rate. If the user first gets nonsense or garbage characters instead of a clear login prompt, he hits the BREAK key causing **getty** to step to the next entry in the sequence. The hunt continues until the baud rate of the line matches the speed of the user's terminal. The flag fields shown have the following meanings:

B300-B19200	The baud rate of the line.
HUPCL	Hang up on close (terminate).
SANE	A composite flag that stands for a set of acceptable line characteristics.
IXANY	Allow any character to restart output. If this flag is not specified, only DC1 (Ctrl-Q) will restart output.
TAB3	Send tabs to the terminal as spaces.

For a description of all **getty** flags, see **termio**(7).

Changing TTY Linesets in /etc/gettydefs

You must be superuser to edit the **/etc/gettydefs** file. In the following example, we will change the login banner for a lineset that starts at M300D and ranges to M1200D (baud range 300-1200). First, before invoking a text editor to make the change, copy **/etc/gettydefs** into **/tmp/gettydefs**. This keeps the original safe until you're ready to finalize your changes. Type

```
# cp /etc/gettydefs  /tmp/gettydefs ↵

# emacs /tmp/gettydefs ↵
```

For the example, we will use the EMACS text editor. Before editing, our file looks like this:

```
M300D# B300 HUPCL DIALOUT SANE IXANY TAB3 CS8 #System Login Banner
DG/UX Release 4.00 login:#M1200D
```

After editing, the file looks like this:

```
M300D# B300 HUPCL DIALOUT SANE IXANY TAB3 CSO #System ZEBRA
DG/UX Release 4.00 login:#M1200D
```

Besides changing `Login Banner` to `ZEBRA`, we accidentally changed `CS8` to `CSO`. But not to worry. Any time you edit **/etc/gettydefs** (or a copy of it as we suggested), you should check the file to see if there are any unrecognized modes or improperly constructed entries. Type

```
# getty -c /tmp/gettydefs  > out ↵
```

The **getty -c** command checks the file then prints out the result of that check in a file named **out** (or whatever you choose). If you look at **out**, you'll see:

```
M300D# B300 HUPCL DIALOUT SANE IXANY TAB3 CSO #System ZEBRA
DG/UX Release 4.00 login:#M1200D

UNDEFINED: CSO
```

After you correct the error, run the **getty -c** command again; then use the **mv** command to write the temporary file over the original:

```
# mv /etc/tmp/gettydefs  /etc/gettydefs ↵
```

End of Chapter

Chapter 11
LP System Management

This chapter shows you how to manage the DG/UX LP system. Originally, LP stood for line printer, but it is now understood to include laser printers as well. Another term often associated with LP systems is spool. The term spool is an acronym for "simultaneous peripheral output online." So when we say LP spooling, we mean that print jobs are queued to be printed on a specific printer, while you continue with other work.

You can manage the LP system simply by executing commands and then responding to the queries that are printed to your screen. When you have a question, type ?.

The major sections of this chapter are:

- LP System Terms
- LP System Procedures
- Printing Path
- LP Directories and Files
- Expert LP Information

LP System Terms

Become familiar with these LP terms before reading on:

scheduler	A program that assigns requests to printer queues. The LP scheduler starts automatically when the DG/UX system comes up.
queue	An ordered list of requests (jobs) to print.
accept	A mode in which a printer will put requests in a queue.
reject	A mode in which a printer will not put requests in a queue.

enable	To allow printing on a given printer.
disable	To stop all printing on a given printer.
device file	A file that represents a physical input/output unit, such as **/dev/tty01**.
model	A prototype printer interface program for sending output to a printer. Six are supplied with the DG/UX system in **/var/spool/lp/model**. These models are **async_1200, async_300, dg455x, lpb, lpj,** and **remshlp**. See the entry for **/var/spool/lp/model** in the section "LP Directories and Files" for more information on these interface programs.

The scheduler program must be running for the LP system to function. You will be asked to start and stop it during the execution of various **sysadm** commands. After a printer has been enabled and placed in accept mode, the scheduler can assign your printing requests to that printer.

LP System Procedures

When you select **lpmgmt** from the Main Menu, the following is displayed:

```
                    Line Printer Management

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

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

The LP system prints error messages when necessary. See Appendix C for a listing and explanation of LP error messages.

Procedure 11.1: Add Printers

Purpose	Add printers to the LP system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**addlp**
References	**lpadmin(1M)**

This command adds a new printer to your system. A printer name may be composed of letters, numbers, and underscores. The name you give a printer is associated with a local or remote printer. For a local printer, one attached to the system you are administering, the name is associated with a device in **/dev** and to an interface script (called a *model*) that sends output to the device. Local printers use model **dumb** by default.

You can "add" a remote printer (one attached to another processor) by specifying the name of a remote system, then specifying the name of the remote printer. Remote printers use model **remshlp** by default. Once added, users on the local system can access the remote printer as though it were local. You should already have installed TCP/IP before you attempt to add a remote printer to your system. Without TCP/IP, you cannot make the remote connection.

To add a printer, select the **addlp** command. We'll add a serial printer. Scheduler messages are printed only if it is already running. The system responds as follows:

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

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

Printer Name? newlp
Is this a local printer? [yes]  ↵
Printer model? [dumb] ↵

Printer device file?  list ↵
The available devices are:

         lp
         tty00 through tty15
```

```
        Printer device file?  tty01 ↵
```

If we had wanted to add a remote printer, say one named *newlp* on another system named *system10,* we would have answered **n** to the local printer query as follows:

```
        Is this a local printer? [yes] n ↵
        Remote host name? system10 ↵
        Remote printer name? newlp ↵
```

Note that for model (printer interface programs), we specified **dumb** for the local case. For the remote case, **remshlp** is automatically used. After this point, whether you're adding a remote or local printer, **sysadm** dialogues are the same.

```
        newlp has been added.
```

Notice that we were asked for a device file (from **/dev**) only for the local printer. For the remote printer, **sysadm** requires the host name and the printer name; no device files are used in the remote case.

```
        Accept and Enable newlp? [yes] ↵

        newlp has been enabled.
        newlp has been set to accept requests.

        Restart the scheduler now? [yes] ↵
        The scheduler has been restarted.

        Press New Line to see the lpmgmt menu [?, ^, q]:
```

Select the **addlp** command again to add another printer.

Procedure 11.2: Delete Printers

Purpose	Remove printers from your system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**dellp**
References	**lpadmin(1)**

Use this command when you are removing a printer from your system, say to install it on another system. Deleting a printer disconnects that printer's name from its association with a device file in **/dev**. For example, if you delete *newlp*, which is associated with **/dev/tty01**, then you are deleting the printer name only. No device files are deleted in **/dev**. After you delete a printer, any attempt to use it will generate an error message.

Make sure that no one is using a printer before you delete it. To delete a printer, select the **dellp** command. The system responds as follows:

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

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

Which printer? newlp ↵
Printer newlp has been deleted.

Restart the scheduler now? [yes] ↵

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Procedure 11.3: Modify an Existing Printer

Purpose	To change the attributes of an existing printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**modlp**
References	lpadmin(1M)

Use this command when you need to change a printer's name or any values associated with it. Defaults shown in the prompts are the previous values that were given to the printer. If you are modifying a remote printer, you must have TCP/IP running.

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

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

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

Printer name? newlp ↵
New Printer Name? [newlp] oldlp ↵
Is this a local printer? [yes]  ↵
Printer model? [dumb] ↵

Printer device file? list ↵
The available devices are:

        tty00 through tty15

Printer device file? [tty01] tty02 ↵
```

If we had wanted to modify a remote printer, say one named *newlp* on another system named *system10*, we would have done the following:

```
Is this a local printer? [yes] n ↵
Remote host name? system10 ↵
Remote printer name? newlp ↵
```

Note that for models (printer interface programs), we specified **dumb** for the local case. **sysadm** uses **remshlp** for the remote case. After this point, whether you're adding a remote or local printer, **sysadm** dialogues are the same.

```
newlp has been modified.
```

Notice that we were asked for a device file (from **/dev**) only for the local printer. For the remote printer, **sysadm** requires the host name and the printer name; no device files are used in the remote case.

```
Accept and Enable oldlp? [yes] ↵

oldlp has been enabled.
oldlp has been set to accept requests.

Restart the scheduler now? [yes] ↵
The scheduler has been restarted.

Press New Line to see the lpmgmt menu [?, ^, q]:
```

Procedure 11.4: List Printers

Purpose	To list printers on the LP system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**lslp**
References	**lpstat(1M)**

This command lists the characteristics of a printer. Characteristics consist of the name, device, enable/disable mode, accept/reject mode, and the length of the print queue.

To list the printers, select the **lslp** command. The system responds as follows:

```
        Which printer? [all]  ↵

Printer Name       Device            Enabled?  Accept?    Requests
------------    ---------------     --------  -------    --------
bigboy          /bigboy@system10    enabled   accepting     2
laser           /dev/tty01          disabled  rejecting     0
newlp           /dev/lp01           enabled   accepting     1

        Press New Line to see the lpmgmt menu [?, ^, q]:
```

Notice the three different `Device` entries. Printer `bigboy` is on a remote processor named `system10` and is not associated with a device file. Printer `laser` is a serial laserprinter, so it is associated with a tty device file. Printer `newlp` is a parallel lineprinter, so it is associated with an **lp** device file.

For information on one printer, type that printer's name.

Procedure 11.5: Set the Default Printer

Purpose	Define the default printer on your system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**defaultlp**
References	**lpadmin(1M)**

This command defines the default printer. This is the printer that is used when someone executes an **lp** command without selecting a particular printer by means of the destination switch, **-d**.

To define a printer as the default, select the **defaultlp** command. The system responds as follows:

```
There is no current default.

New default printer?  newlp ↵

The new default printer is: newlp.

Press New Line to see the lpmgmt menu [?, ^, q]:
```

Procedure 11.6: Set a Printer to Accept Mode

Purpose	To put a printer into accept mode.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**acceptlp**
References	**accept(1M)**

In this mode, the printer will accept and queue all requests. You may want to disable a printer, but leave it in accept mode, when the printer will be down for a short time. This way, the printer queue will still accept job requests. If you select **acceptlp**, the system responds as follows:

```
Which printer (for accept)? [newlp] ↵

Printer newlp has been set to accept requests.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Now, printer *newlp* will accept jobs submitted to it, that is, jobs will go into a queue associated with *newlp*. If it has not been put in accept mode, **sysadm** prints a message telling the user what steps must be taken.

Procedure 11.7: Set a Printer to Reject Mode

Purpose	To put a printer into reject mode.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**rejectlp**
References	**accept(1M)**

Use this command to reject any new print requests. Once placed in reject mode, the printer will continue printing whatever has been queued. When users attempt to make requests to a printer that has been put in reject mode, a message will be displayed explaining why the printer is not accepting new print requests. You might use this command when a very large job is printing and you want to divert users elsewhere. Or, you might use this command when you know that a printer will be down long enough to inconvenience users.

If you select the **rejectlp** command, the system responds as follows:

```
Which printer (for reject)? [newlp] ↵

Please give a one-line reason for rejecting requests
on printer newlp.

Reason? Printer newlp is down for repairs.

Printer newlp has been set to reject requests.

Press the New Line key to see the lpmgmt menu [?,^,q]:
```

Procedure 11.8: Start and Stop Printers

Purpose	To start and stop a printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**enablelp** **disablelp**
References	**enable(1)**

Enable an LP

After you enable a printer, it is available to begin printing on command. But if the printer is in reject mode, it will not allow users to submit print requests. If you select the **enablelp** command, the system responds as follows:

```
Which printer (for enable)? [newlp] laser ↵

Printer laser has been enabled.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

If you want to see the list of available printers, type ?.

Disable an LP

If a printer is disabled, it will not print any jobs in the queue. The printer *will* continue to accept jobs (queue them). When users attempt to use a disabled printer, a message is displayed explaining why the printer is disabled. If you select the **disablelp** command, the system responds as follows:

```
Which printer (for disable)? [newlp] ↵

Please give a one-line reason for disabling printer newlp.
Reason?  Printer is very busy right now.

Printer newlp has been disabled.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

The one-line reason will be displayed when someone tries to use the printer.

Procedure 11.9: List Print Job Queue

Purpose	To display print requests (jobs) in the queue.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**queuelp**
References	**lpstat(1)**

This command displays the list of print requests that are waiting to be sent to a printer. The first line shows the next job to be printed. Individual requests are numbered and attached to the printer name.

To list print requests (jobs) in the queue, select the **queuelp** command. The system responds as follows:

```
    Which printer (to list)? [newlp] ↵

    Printer newlp:

    Request                User     Size     Time
    ---------             -----     ----    -----------
    newlp-11                root     17      Jul 27 18:23
    newlp-12            robocop     4224     Jul 27 18:30
    newlp-13               tobor     2345     Jul 27 18:33

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Here, we chose to see the queue of the default printer, `newlp`. Typing **all** will list the queues of all printers.

Procedure 11.10: Cancel a Job

Purpose	To cancel a job in the LP system queue.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**cancellp**
References	**lp(1), cancel(1)**

This command cancels a job in the queue. The job may be printing or waiting to print. If you cancel a job while it is printing, it will be stopped, and the next job will begin printing.

To cancel a job in the queue, select the **cancellp** command. The system responds as follows:

```
  Which printer (for cancel)? [newlp] ↵

 The current requests on newlp are:

 Request                User     Size      Time
 ----------             -----    ----   ------------
 newlp-11                root    17     Jul 27 18:23
 newlp-12             robocop    4224   Jul 27 18:30
 newlp-13               falco    2345   Jul 27 18:33

 Which request(s) should be canceled? [none] newlp-11 ↵

The requests have been canceled.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Procedure 11.11: Move Jobs to New Printer

Purpose	To move jobs in one printer's queue to the queue of another printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**movelp**
References	**lpsched(1M)**

This command moves print requests from the queue of one printer to that of another printer. **Movelp** is useful when one printer must be taken out of service and another printer is available to handle its load. **Movelp** will move all or some requests. If all are moved, it will place the printer with the emptied queue in the reject mode.

To move requests from one printer to another, select the **movelp** command. The system responds as follows:

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

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

From printer? newlp ↵
To printer? bigboy ↵

The current requests on newlp are:

Request              User      Size      Time
---------           -------    ----    -----------
newlp-12            robocop    4224    Jul 27 18:30
newlp-13              tobor    4550    Jul 27 18:33

Which request? [none]  all ↵

The requests have been moved.
```

```
Because you have moved all requests from newlp to bigboy,
sysadm has left newlp in the reject mode.  All attempts to
use newlp will fail until you use acceptlp to change the mode.
```

If you used the **queuelp** command for printer bigboy, you'd see that the jobs have been moved.

```
Restart the scheduler? [yes] ↵
The scheduler has been restarted.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Procedure 11.12: Start and Stop the LP Scheduler

Purpose	To start and stop the LP scheduler, the program that sends print requests to the printers. If the scheduler is stopped, the entire LP system stops.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**startlp** **stoplp**
References	**lpsched(1M), lpshut(1M)**

These commands start and stop the LP scheduler, the program that actually sends print requests to the printers. When the scheduler is not running, the entire printer subsystem is not running.

Use the **stoplp** command to stop the scheduler to perform administrative tasks. When you're ready to start up again, use the **startlp** command. Depending on which command you choose, the system responds in one of the following ways:

```
The lp scheduler has been stopped.
```

or

```
The lp scheduler is now running.

Press the New Line key to see the lpmgmt menu [?, ^, q]:
```

Printing Path

Figure 11-1 shows the path of a file named "memo" as it goes through the LP system and is printed.

$$\begin{tikzcd}\text{memo} \arrow[r, "\textbf{lp -dlaser}"] & \text{queue (spool file)} \arrow[r, "\text{lpsched to interface}"] & \text{printer}\end{tikzcd}$$

Figure 11-1 LP System Print Path

The steps are:

1) A user includes the **-d** option with the **lp** command to specify that memo be printed on a printer named laser.

2) The **lp** command puts the request in the queue, **/var/spool/lp/outputq**.

3) The **lpsched** program reads the request from the queue file and passes it on to the interface program.

4) The interface program sends the request to the specified printer and the file memo is printed.

LP Directories and Files

This section describes the files and directories in the LP system.

/var/spool/lp/class

Optional. This is a directory that contains one file for each LP class that has been identified. (The name of the file is the same as the name of the class.) The file identifies each member, in this case an LP printer, that is assigned to the class. Class files are created, modified, and deleted by the **lpadmin** command. A class is a group of printers that you have named such for reasons of performance or location, or for whatever. For instance, you might have a class named LPB1. In class LPB1 are printers LPB2 and LPB3. Requests to print go to the class, then the least busy printer prints the job request.

/var/spool/lp/FIFO

FIFO is a special file that all the commands use to send messages to **lpsched**. Any of the LP commands may write to **FIFO**, but only **lpsched** may read it.

/var/spool/lp/default

This file contains the name of the system default destination printer. If this file does not exist or if it is empty, the LP system has no default printer.

/var/spool/lp/interface

The **interface** directory contains one executable interface script for each printer that is in the LP system. The filename of the interface script is the same as the printer name. The interface program is invoked with its standard error and standard output directed to the printer. Interface programs may be shell procedures or compiled C programs.

/var/spool/lp/log

The purpose of the **log** file is to keep a record of all the printing activity that has taken place since the LP scheduler was last started. This file contains the logname of the user who made the request, the request ID, the name of the printer that the request was printed on, and the date and time that printing started. Any **lpsched** error messages that occur are also recorded. The first line of the log file shows the time that the LP scheduler was started.

/var/spool/lp/member

The **member** directory contains one file for each LP printer. The filename is the same as the printer name. Each file contains the pathname of the device to which the member is connected.

/var/spool/lp/oldlog

The **oldlog** file contains a record of what was in the previous **log** file. When the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, all the information that had accumulated in the **log** file is copied to the **oldlog** file, and a new **log** file is started. Any information that had been in the **oldlog** file is overwritten. The first line of the file tells the time that the scheduler was turned on, and the last line tells the time the scheduler was turned off.

/var/spool/lp/outputq

Basically, **lp** writes to this binary file and the LP scheduler reads that information from this file. It works like this. When an output request is made via the **lp** command, an entry is made in **outputq**. The LP scheduler takes the job request and passes it to the appropriate interface program to be printed. When the job is completed, a binary tag is attached to that job entry. Also, whenever the commands **lpmove**, **disable**, **lpsched**, and **cancel** are executed, the binary tag is attached to the job in question. This tag tells the scheduler not to run the job again.

/var/spool/lp/pstatus

The binary file **pstatus** contains status information for each printer. Entries are added to and removed from this file by the **lpadmin** command, and they are modified by the **cancel**, **enable**, **disable**, and **lpsched** commands. When the **lpstat** command is invoked with the **−p** option, printer status information is obtained from this file.

/var/spool/lp/qstatus

This binary file keeps track of whether a destination printer is accepting or rejecting requests. Entries are added or removed from this file by the **lpadmin** command and modified by the **accept** and **reject** commands. When the **lpstat** command is invoked with the **−o** option, the request status is obtained from this file.

/var/spool/lp/seqfile

The **seqfile** file contains the sequence number of the last request ID that was assigned by the **lp** command. The sequence number (**seqno**) is incremented by **lp** for each request. When the number 10101010 is reached, the sequence number is reset to 1.

/var/spool/lp/model

This is a directory that contains the printer interface programs that are distributed with the DG/UX system. These programs are:.

async_1200	For asynchronous line printer. Sets line to 1200 baud, x-on/x-off protocol. Tabs are expanded before printing.
async_300	Same as above, but sets the baud rate to 300, 600, 2400, 4800, or 9600.
dg455x	For Data General Model 4557 or 4558 laser printer. This interface supports two options: **-o66** 66 lines per 11-inch page -- compressed vertical spacing **-o62** 62 lines per 11-inch page -- normal 6 lines per inch
lpb	For Data General data channel line printer for commercial I/O subsystems that need format loading; supports model numbers 4215-19 and 4244-45. This interface program loads tape format before a file is printed at each invocation.
lpj	For Data General asynchronous line printers. Sets the line to 2400 baud, x-on/x-off protocol. Tabs are expanded before printing.
remshlp	For accessing remote printers over a network. Executes the **lp** command on a remote system through a remote shell (remsh).

Administrators may also add their own interface programs to this directory.

/var/spool/lp/request

This directory contains a subdirectory for each destination in the LP system. The name of the subdirectory is the same as the name of the destination. When an **lp** request is made, a **request** file (or "r" file) and, in most cases, a **data** file (or "d" file) are created in the subdirectory of the destination to which the request is going. The data file stores the file to be printed until the scheduler is ready to print it. A data file is not created if the file to be printed cannot be linked to the request subdirectory.

The name of the request file is derived from the request identification number and is of the form **r-seqno**. The name of the data file is of the form **dn-seqno**, where **n** is a non-negative integer.

The request and data files are deleted by the **cancel** and **lpsched** commands. They may be moved from one subdirectory to another by the **lpmove** command.

Lock Files

To guarantee LP commands exclusive access to data files, several lock files are maintained in the LP system. They are binary files that contain the process ID of the locking process. The lock files and their associated data files are the following:

Lock File	Data File
OUTQLOCK	outputq
PSTATLOCK	pstatus
QSTATLOCK	qstatus
SEQLOCK	seqfile

Lock files "expire" after ten seconds and may be unlinked by any LP process. If a file is older than 10 seconds (not active), then the next process will begin. Thus, commands that lock a data file for longer than this interval must update the modification time on the lock file. The creation, updating, and unlinking of lock files is handled automatically by the LP system. Another lock file, SCHEDLOCK, is present while the LP scheduler is running to ensure that only one invocation of **lpsched** is active. Unlike other lock files, SCHEDLOCK has no expiration time.

Cleaning Out Log Files

As described above, when the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, the **log** file is copied to **/var/spool/lp/oldlog**, and a new **log** file is started.

If the scheduler is not stopped for long periods of time and if you have a large number of LP requests, the **log** file can grow to be a large file. You can manually remove the contents of this file, or you can let the system do it for you on a scheduled basis. To have the system clean out the log file periodically, put an entry in a file in the **/var/spool/cron/crontabs** directory. Log in as **root** and use the **crontab(1)** command.

The example below shows some typical **crontab** command lines. The **crontab** command adds these lines to the **root** file in the **/var/spool/cron/crontabs** directory. Every Friday at 11:00 PM **cron(1M)** executes the commands. First, the contents of the **log** file are copied to the **oldlog** file, and then the **log** file is cleaned out. So, you have a small, manageable file for the current week, and you can look back and read the file for the previous week.

```
# crontab -l

0 23 * * 5 /bin/su lp -c "cp /var/spool/lp/log /var/spool/lp/oldlog"
1 23 * * 5 /bin/su lp -c ">/var/spool/lp/log"
```

Expert LP Information

The DG/UX system comes with printer interface scripts for all Data General equipment. If you need to write your own interface script, you should read the following sections. Otherwise, the information in the following sections is optional. We supply it for administrators who want to know more about the system. You do not need to read these sections to administer the LP system.

Administrative Commands

Table 11-1 shows a separate set of commands available for the LP administrator. These commands are in the **/usr/lib** directory. If you expect to use them frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in as **root** or **lp**.

Table 11-1 Administrative Commands for the LP System

Command	Description
/usr/lib/accept	Permit job requests to be queued for a specified destination.
/usr/lib/reject	Prevent jobs from being queued for a specified destination. Described on the same manual page as **accept(1M)**.
/usr/lib/lpadmin	Set up or change the LP configuration.
/usr/lib/lpmove	Move output requests from one destination to another. Described on the same manual page as **lpsched(1M)**.
/usr/lib/lpsched	Start the LP scheduler.
/usr/lib/lpshut	Stop the LP scheduler. Described on the same manual page as **lpsched(1M)**.

Command Descriptions and Examples

/usr/lib/lpadmin

The **lpadmin(1M)** command is used to add a new printer to the system, assign classes of printers, name or remove a default destination, and specify interface programs to be used. **lpadmin** may not be used when the LP scheduler, **lpsched(1M)**, is running, except when the **−d** option is specified.

You must include one (and only one) of the following three options on the **lpadmin** command line:

−d[*dest*] Defines an existing system destination as the new default destination. If *dest* is not specified, there is no default destination. The LP scheduler may be running when you use this option. The default destination is used to determine where a file named in a user's **lp** command is sent. The destination (*dest*) must already exist.

−x*dest* Removes a destination (*dest*). You cannot invoke this option when the scheduler is running. If it is running, you must issue the **lpshut** command before **lpadmin**.

No destination (class or printer) may be removed if it has pending requests. The pending requests must either be cancelled using the **cancel(1)** command or moved to other destinations using the **lpmove(1M)** command before the destination can be removed.

Removing the last remaining member of a class causes the class to be deleted. If the destination removed is the system default destination, the system will no longer have a default destination. However, the removal of a class does not imply the removal of printers that were assigned to that class.

−p*printer* Names a printer to which arguments apply. If *printer* does not exist, it is created.

No other options are allowed with **−d** and **−x**. However, many arguments are allowed with the **−p***printer* option, and at least one argument must always be present. The arguments that can be used with **−p** are as follows:

−c*class* Assigns the printer specified in the **−p** option to the specified *class*.

−e*printer* Allows you to use an existing interface program for a new printer that you are adding to the LP system. When you select

	this argument, the interface program for the printer specified in this argument is copied for the printer specified in the **−p** option.
−i*interface*	Specifies a new interface program for the printer specified in the **−p** option. *interface* is the path name of the new program.
−l	Indicated that, when adding a new printer, the device associated with the printer is a login terminal.
−r*class*	Removes a printer from the specified class.
−v*device*	This argument must be used when you add a new printer to the LP system. It associates the printer with the device file specified by *device*. The complete path name must be given for the file.

Command Examples

In examples 2 through 7, the LP scheduler has already been stopped with the **lpshut(1M)** command. Example 1 does not require the scheduler to be stopped since only the **−d** option to **lpadmin** is used.

Example 1

Make printer *newlp* the system default destination.

```
# lpadmin −dnewlp ↵
```

Example 2

Add a new printer called *fastlp* and associate it with device **/dev/tty11**. Use the **aslp** model interface program.

```
# lpadmin −pfastlp −v/dev/tty11 −maslp ↵
```

When you add a new printer, it is left in a disabled state and does not accept requests.

Example 3

Create a hardwired printer called *lp1* on device **/dev/tty13**. Add *lp1* to a new class called *cl1*, and use the same interface program that is used with printer *fastlp*.

```
# lpadmin −plp1 −v/dev/tty13 −efastlp −ccl1 ↵
```

Example 4

Change the interface program for printer *lp1* to model interface program **dclp**.

```
# lpadmin -plp1 -mdclp ↵
```

Example 5

Add printer *fastlp_2* to class *cl1*:

```
# lpadmin -pfastlp_2 -ccl1 ↵
```

Printers that you add to a class are ordered according to the sequence in which you add them. If all three printers are available, and a request is routed to class *cl1*, the request will be serviced by the one that you added first. If all three printers are busy, the request will be printed by the first available printer.

Example 6

Remove printers *newlp* and *fastlp* from class *cl1*:

```
# lpadmin -pnewlp -rcl1 ↵
# lpadmin -pfastlp -rcl1 ↵
```

/usr/lib/lpsched

The LP scheduler starts automatically each time the system is booted. It does so via an **rc** script named **rc.lpsched**.

The **lpsched(1M)** command starts the LP scheduler. The LP scheduler takes the top job request off the queue and "hands" it to the appropriate interface program to be printed on a printer. The LP scheduler keeps track of the job progress, and as soon as the job is completed, it takes the next job request off the queue and repeats the process. As long as the LP scheduler is running, jobs requested by **lp** will be printed. If the scheduler is not running, jobs will not be printed.

Every time the scheduler is started, **lpsched** creates a file called SCHEDLOCK in the **/var/spool/lp** directory. As long as the SCHEDLOCK file is present, the system will not allow another scheduler to run. When the scheduler is stopped under normal conditions, either with **lpshut(1M)** or as part of the normal shutdown procedure, the SCHEDLOCK file is removed. However, if the system comes down abnormally, there is a possibility that the SCHEDLOCK file may not get removed. To ensure that the SCHEDLOCK file does not exist, **/etc/rc.d/lp** contains a command line to remove SCHEDLOCK first before it attempts to start the scheduler.

Type the command without arguments as follows:

```
# lpsched ↵
```

Note that the command shows no response to let you know that the scheduler is running. To verify that the scheduler is running, use the **lpstat(1M)** command with the **−r** option.

```
# lpstat −r ↵
  scheduler is running
```

/usr/lib/lpshut

Two of the three **lpadmin** command options (**−x** and **−p**) cannot be executed unless the LP scheduler is stopped. The **lpshut** command stops the LP scheduler and terminates all printing activity. All requests that were in the middle of printing will be reprinted in their entirety when the scheduler is restarted. Type this command without arguments as follows:

```
# lpshut ↵
  scheduler stopped
```

/usr/lib/lpmove

Occasionally, you may find it necessary to move output requests from one destination to another. For example, if you have a printer that was removed for repairs, you will want to move all the pending job requests to a destination with a working printer. This is done using the **lpmove** command. Be aware that job requests routed to a destination without a printer are automatically rejected.

Another use of the **lpmove** command is to move specific requests from one destination to another. When this is done, **lp** will no longer accept requests for the original destination (this is the same effect as a **reject** command). **lpmove** refuses, however, to move requests while the LP scheduler is running. The general format of the **lpmove** command is as follows:

lpmove *requests dest*

requests are the request identification numbers (request IDs) of jobs waiting to be printed, and *dest* is the destination to which the requests are to be moved. The destination can be a printer or a class of printers.

Command Examples

The following examples show how you can use **lpmove**:

Example 1

Move all the requests for printer *lp1* to printer *lp2*. Moving the requests renames the request IDs from *lp1-nnn* to *lp2-nnn*. After the requests are moved, **lp** will no longer accept requests for *lp1* (this is the same effect as a **reject lp1** command issued after the **lpmove**).

```
    # lpmove lp1 lp2 ↵
```

Example 2

Move requests *lp1-11* and *lp2-25* to printer *newlp*:

```
        # lpmove lp1-11 lp2-25 newlp
          total of 2 requests moved to newlp
```

/usr/lib/accept

The **accept**(1M) command allows job requests to be placed in a queue at the named destination(s), destination being the name of a printer or class of printers. The general format of the **accept** command is as follows:

> **accept** *destination(s)*

Command Example

The sample command line allows printer *newlp* to start receiving requests.

```
        # /usr/lib/accept newlp ↵
           destination "newlp" now accepting requests
```

/usr/lib/reject

Sometimes it is necessary to stop **lp** from routing requests to a destination. For example, if a printer has been removed for repairs, or if too many requests are building at a destination, you may want to prevent new jobs from being queued at this destination. The **reject(1M)** command performs this function.

Requests in the queue when the **reject** command is invoked will be printed as long as the printer is enabled. After the condition that led to denying requests has been corrected, use the **accept** command to allow requests to be received again. The general format of the **reject** command is as follows:

> **reject** [**−r**[*reason*]] *destinations*

The **−r** option is for telling users why requests are being rejected. The *reason* is a brief explanation of the purpose for rejecting requests. If the reason consists of more than one word, enclose it in double quotes. The *destinations* are the printers that will no longer accept requests.

Command Example

The example given here is for a printer, *fastlp*, that is being repaired. While it is out of service you want to prevent **lp** from routing requests to it.

```
# reject -r "printer fastlp under repair" lqp40_1
  destination "lqp40_1" is no longer accepting requests
```

Users who try to route a job to *fastlp* will receive the following message:

```
$ lp -dfastlp file1 ↵
  lp: can't accept requests for destination "fastlp" -
  printer fastlp under repair
```

Printer Interface Scripts

Printers must have an interface script to work with the DG/UX system. Every print request made with the **lp** command is routed through the appropriate printer interface script before the request is printed on a line printer. Use the **lpadmin -i** command to associate a printer with an interface program. See **lpadmin(1M)**.

Model Interface Scripts

Each type of printer requires its own interface script. Several prototype interface scripts, called models, are furnished with the DG/UX system. The model interface scripts support the LPB printer and the LPJ printer. The model interface scripts are written as shell procedures, but they can be written as C programs or any other executable program. The DG/UX system uses copies of the models. They are located in the **/var/spool/lp/model** directory.

Writing Interface Scripts

If you have a printer that is not supported by one of the model programs, you will have to furnish an interface script for it. The shell script for a "dumb" printer interface script (a model program) is shown at the end of this subsection. This program may be used as a guide if you have to do one of your own.

When the LP scheduler routes an output request to a printer, the interface script for that printer is invoked in the directory **/var/spool/lp** as follows:

interface/*P ID user title copies options file*

Arguments for the interface program are:

P	Printer name

id	Request ID returned by lp
user	User who made the request
title	Optional title specified by the user
copies	Number of copies requested by user
options	Blank-separated list of options specified by user
file	Full path name of a file to be printed

When the interface program is invoked, its standard input comes from **/dev/null** and both the standard output and standard error output are directed to the printing device. Interface programs format their output based on the command line arguments. Make sure that the interface program has the proper stty modes (terminal characteristics such as baud rate, output options). You can do this by adding **stty(1)** command lines of the form:

```
# stty mode options <&1
```

This command line takes the standard input for the **stty** command from the device. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below.

```
# stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by **lpsched** as follows:

Code	**Meaning to lpsched**
0	The print job has completed successfully.
1 to 127	A problem was encountered in printing this particular request (for example, too many nonprintable characters). This problem will not affect future print jobs. The **lpsched** command notifies users by **mail(1)** that there was an error in printing the request.
greater than 127	These codes are reserved for internal use by **lpsched**. Interface programs must not exit with codes in this range.

Dumb Line Printer Interface Program

The following program is provided as a guide should you need to write one of your own.

```
# Copyright (C) Data General Corporation, 1984 - 1988
# All Rights Reserved.
# Licensed Material-Property of Data General Corporation.
#
# This software is made available solely pursuant
# to the terms of a DGC license
# agreement which governs its use.
#
# "@(#)dumb 9.2"
#
#      dumb - Line Printer Interface Script for dumb asynchronous line printer
#
#model# Dumb Asynchronous Line Printer (no baud rate)
#
# PARAMETERS:
#           This interface is called with the following parameters:
#
#           $0 - interface name     - "interface/'printer_name'"
#           $1 - Request id returned by lp
#           $2 - Logname of user making request
#           $3 - Optional title specified by user
#           $4 - Number of copies specified by user
#           $5 - Blank seperated list of options - none apply to this interface
#           $6+- Filenames to be printed
#
# EXIT CODES:
#           This interface returns codes that are interpreted by "lpsched"
#           as follows:
#           0        - Print job was completed successfully
#           1 to 64  - A problem code that does not affect future jobs
#           1            - Bad options list
#
#           65 - 127 - If the problem detected affects future jobs, the
#                        printer will be disabled. Exit codes are then:
#           65           - Can not execute filter - /usr/lib/lptab
#
#           over 127 - Reserved for internal use by lpsched.
#

# check options - none apply to this printer interface
if [ ! -z "$5" ]
then
            exit 1
fi
# check if filter is executable
if [ ! -x /usr/lib/lptab ]
then
            disable -r"Can not execute /usr/lib/lptab" `basename $0`
            exit 65
fi

stty tabs opost onlcr -onlret clocal ixon ff1 cr2 nl0 0<&1

x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
echo " 14echo "$x0
banner "$2"
echo "0
user=`grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ]
then
            echo "User: $user0
else
            echo "0
fi
echo "Request id: $1    Printer: `basename $0`0
date
echo "0
if [ -n "$3" ]
then
            banner $3
fi
copies=$4
echo " 14shift 5
files="$*"
i=1
while [ $i -le $copies ]
do
            for file in $files
            do
                        /usr/lib/lptab < $file 2>&1
                        echo " 14           done
            i=`expr $i + 1`
done
echo "$x"
exit 0
```

Figure 11-2 Sample LP Interface Program

End of Chapter

Chapter 12
UUCP Management

The DG/UX System uses the HoneyDanBer version of UUCP. You can find additional expert information in Appendix E. For a listing of UUCP error messages, see Appendix C.

The major sections of this chapter are

- What is UUCP?
- UUCP Setup Overview
- UUCP Programs, Daemons, and Data Files
- UUCP Directories
- Remedies for Common UUCP Problems
- UUCP Management Procedures

What is UUCP?

We refer to uppercase UUCP and lowercase **uucp**; both originate from UNIX-to-UNIX Copy. Lowercase **uucp** refers to a DG/UX command, **uucp(1)**. UUCP refers to a set of programs and data files that allow you to transfer files and to execute remote commands between UNIX systems. Throughout this chapter, we'll mainly be referring to the set of programs; when we refer to the command, we'll use the **(1)** notation. The functions of UUCP are also used by the **mail(1)** program for remote exchanges.

When we talk about UUCP connections, we mean via a *direct* or a *dial-up* connection. A direct connection is simply a physical wire between machines. A dial-up connection uses telephone lines and a modem at either end to connect machines.

UUCP Components

UUCP works because of the interaction of multiple files and programs. These are briefly discussed in this chapter and are more thoroughly discussed in Appendix E. The five main **/usr/lib/uucp** files are: **Systems, Poll, Devices, Dialers,** and **Permissions**. You can only connect 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 **Poll**. 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 **Devices**. The entries in **Devices** need data from **Dialers**. Finally, the **Permissions** file restricts a remote computer's ability to request and receive files. The default **Permissions** file is set up to provide the maximum amount of security. You can use the **uucheck -v** command to see exactly what your default permissions are. If you wish to change them, see "Permissions File" in Appendix E.

Before Using UUCP

Before you can put this system to work, you must have a location with which you wish to set up file transfer connections. This means you will have to contact the system administrator of a remote site and exchange certain information: passwords, system NODE names, baud rates, and phone numbers. When you have exchanged this information, you are ready to set up the files.

UUCP Setup Overview

We recommend that you setup your UUCP facility in the following order:

1) Read "HoneyDanBer: New UUCP vs Old UUCP"
2) Start the **uudemon.poll, uudemon.hour, uudemon.admin,** and **uudemon.cleanup** shell scripts.
3) Add devices with **adddevice**.
4) Add systems with **addsystem**. Also edit **/usr/lib/uucp/Systems** and **/etc/inittab** for systems running **uugetty**.
5) Add poll entries with **addpoll**.
6) Test your connections with **trysystem**.

HoneyDanBer: New UUCP vs Old UUCP

There are two versions of UUCP: a new version, referred to as HoneyDanBer, and an old version. Some files have been renamed; we list them in "UUCP Data Files" later in this chapter. Functional differences between the two versions of UUCP are reflected in the **/etc/inittab** file. The new version allows for bidirectional login by respawning **uugetty** instead of **getty**. Bidirectional means that a computer can call *or* receive. So if you have new UUCP on your system, you can connect with other systems running old UUCP. To do this, the **inittab** and **Systems** files must be set up correctly. After using **sysadm addsystem**, you'll need to do some editing to reflect what versions of UUCP you'll communicate with; see "Connecting Like and Unlike Versions of UUCP" later in this chapter.

Starting the UUCP Shell Scripts

Your first step in setting up UUCP is to start several important shell scripts. We'll explain about these scripts, then show you how to start them.

uudemon.poll

- Reads the **Poll** file (**/usr/lib/uucp/Poll**) as scheduled.
- If any of the machines in the **Poll** file are scheduled to be polled, a work file (C.*filename*) is placed in the **/var/spool/uucp/***nodename* directory, where *nodename* is replaced by the name of the machine.

uudemon.hour

- Calls the **uusched** program to search the spool directories for work files (of the form C.*filename*) that have not been processed and schedules these files for transfer to a remote machine.
- Calls the **uuxqt** daemon to search the spool directories for execute files (of the form **X.***filename*) that have been transferred to your computer and were not processed at the time they were transferred.

uudemon.admin

- Runs the **uustat** command with **−p** and **−q** options. The **−q** reports on the status of work files (C.*filename*), data files (**D.***filename*), and execute files (**X.***filename*) that are queued. The **−p** prints process information for processes listed in **/var/spool/locks**.
- Sends resulting status information to the **nuucp** administrative login via **mail(1)**.

uudemon.cleanup

- Takes log files for individual machines from the **/var/spool/uucp/.Log** directory, merges them, and places them in the **/var/spool/uucp/.Old** directory with other old log information.
- Removes work files seven days old or older, data files seven days old or older, and execute files two days old or older from the spool files.
- Returns mail that cannot be delivered to the sender.
- Mails a summary of the status information gathered during the current day to the **nuucp** administrative login.

The following lines are **cron** instructions to run the scripts listed above. Edit these into a file of your choice. Below, we created the file **uuscripts** and put in the following lines:

```
1,30 * * * * "/usr/lib/uucp/uudemon.poll > /dev/null"
41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
48 22 * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.admin" >
/dev/null
45 23 * *  ulimit 5000; /bin/su uucp -c "/usr/lib/uucp/uudemon.cleanup"
> /dev/null 2>&1
```

Next log in as **nuucp** and type the following:

```
# cd /usr/lib/cron ↵
# echo > cron.allow ↵
```

We created **cron.allow**. Now, use an editor and add **nuucp** to this empty file; this ensures that **nuucp** will have the correct permissions to function with the **cron** program. Next type

```
# su nuucp
# crontab uuscripts
```

Your UUCP shell scripts are started and will run periodically as specified above. To change the schedules set above, simply edit the lines and run **crontab(1)** on **uuscripts**. Now you're ready to set up the rest of UUCP with **sysadm uucpmgmt**.

UUCP Programs, Daemons, and Data Files

This section discusses the major components of UUCP. See Appendix E for more information.

Administrative Programs

The following programs are available to be used at your option.

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**; computers send this login name to start communications. Calls are answered by **uucico**. See Chapter 4 for more information on administrative logins.

uuname	Lists those machines you can contact. This command is in **/usr/bin**. You can do this from the command line or you can use **sysadm lssystem**.
uulog	Displays the contents of the log directories for specified hosts. Log files are created for each remote computer your computer communicates with. The log files contain records of 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. It is normally executed from a shell script called **uudemon.cleanup**, which is started by **cron**.
Uutry	Tests connections between computers. This command displays messages on failed or successful sessions. It also does a moderate amount of debugging. It invokes the **uucico** daemon to establish a communication link between your computer and the remote computer you specify. You can also do this with the **sysadm trysystem** command.
uucheck	Checks for the presence of UUCP directories, programs, and support files. With a **-v** option, it displays the current permissions for your system. This command is in **/usr/lib/uucp** and is not available through **sysadm**.

User Programs

The user programs for UUCP are in **/usr/bin**. No special permission is needed for these programs. These commands are described in the *User's Reference for the DG/UX System*.

ct	This program instructs your computer to call a modem attached to a terminal over the telephone network. When the modem answers, your computer issues a **getty** (login) process to the modem and allows the terminal user to log in. The user of the remote terminal may call in to the computer and request that the computer call the remote terminal back. The computer will hang up the initial link to the terminal so that it will be available to answer the call back. This is similar to making a collect call.
cu	This program connects your computer to a remote computer and allows you to be logged in on both computers at the same time. You can execute commands on either computer without dropping the communication link.
uucp	Lets a user copy files from one computer to another. It creates work files and data files, queues the job for transfer, and calls the **uucico** daemon, which in turn attempts to contact the remote computer.
uuto	Copies files from one computer to a public spool directory on another computer (**/var/spool/uucppublic/receive**). Unlike **uucp**, which lets you copy a file to any accessible directory on the remote computer, **uuto** places the file in an appropriate spool directory and tells the remote user to pick it up with **uupick**.
uupick	Retrieves the files placed under **/var/spool/uucppublic/receive** when files are transferred to a computer using **uuto**.
uux	Creates the work, data, and execute files needed to execute commands on a remote computer. The work file contains the same information as work files created by **uucp** and **uuto**. The execute files contain the command string to be executed on the remote computer and a list of the data files. The data files are those files required for the command execution.
uustat	Displays the status of requested transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Daemons

Daemons are routines that run as background processes and perform system-wide public functions. These daemons handle file transfers and command executions. They can also be run manually from the shell.

uucico Selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and execute files, logs results, and notifies the user by mail of transfer completions. When the local **uucico** daemon calls a remote computer, it "talks" to the **uucico** daemon on the remote computer during the session.

The **uucico** daemon is executed by **uucp**, **uuto**, and **uux** programs, after all the required files have been created, to contact the remote computer. It is also executed by the **uusched** and **Uutry** programs.

uuxqt Executes remote execution requests. It searches the spool directory for execute files (always named **X.***file*) that have been sent from a remote computer. When an **X.***file* file is found, **uuxqt** opens it to get the list of data files that are required for the execution. It then checks to see if the required data files are available and accessible. If the files are present and can be accessed, **uuxqt** checks the **Permissions** file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed by the **uudemon.hour** shell script, which is started by **cron**.

uusched Schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote computers will be called. **uusched** is executed by a shell script called **uudemon.hour**, which is started by **cron**.

UUCP Data Files

The support files for UUCP are in the **/usr/lib/uucp** directory. You can make all changes to these files with **sysadm uucpmgmt**. Below, we provide details on the structure of these files in case you want to edit them manually. Because release 4.0 of the DG/UX operating system uses a new version of UUCP, the support files are named differently. Below, we list the names of these for earlier releases of the DG/UX system.

Devices Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices. This file was previously called **L-devices**.

Dialers	Contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers (non 801-type dialers). This is a new file that contains some sample chat scripts. If your dialer is not already in this file, add it.
Systems	Contains information needed by the **uucico** daemon and the **cu** program to establish a link to a remote computer. It contains information such as the name of the remote computer, the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login ID, and password. This file was previously called **L-sys**.
Dialcodes	Contains dial-code abbreviations that may be used in the phone number field of **Systems** file entries. This file was previously called **L-dialcodes**.
Permissions	Defines the level of access that is granted to computers when they attempt to transfer files or remotely execute commands on your computer. Previously, this file was split into **USERFILE** and **L.cmds**.
Poll	Defines computers that are to be polled by your system and when they are polled. This is a new file.
Sysfiles	This file is used to assign different or multiple files to be used by **uucico** and **cu** as **Systems**, **Devices**, and **Dialers** files. This is a new file.

UUCP Directories

/usr/bin
Used by the DG/UX operating system to store UUCP programs.

/usr/lib/uucp
The HOME directory for the **uucp** login. It contains the files that make up the supporting database.

/var/spool/locks
Contains lock files for UUCP devices.

/var/spool/uucp
The spool directory for queued work that is to be processed by UUCP daemons. Also contains directories for administrative purposes and for storing log and status information.

/var/spool/uucppublic
The public directory for UUCP. Stores work that has been sent to your computer.

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: check the expect/send sequence in the **Systems** file.
- Remote system name mismatch: make sure the system name in the **Systems** entry matches the nodename of the remote system. On the remote system, use the **uname -n** command to get the nodename.

Modem cannot make connection

- Check modem switches (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

- Check modem switches.
- Put a phone on the line and call it.
- Make sure **getty** or **uugetty** is up.
- If running **uugetty**, try typing anything or press carriage return.

uucico or cu dies immediately	• Use appropriate debugging switch first. If you get no response or a hangup, make sure **Systems, Devices, Permissions,** and **Dialers** files are present and readable by **uucp**. • Make sure **passwd** and **group** files are correct.
Hung modem	Send the reset command (such as ATZ for Hayes modems) to the modem.
Hung syac	Reload the syac with **syacload**.
Modem in wrong state	Such as echo mode, where letters appear twice on the screen: check 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.
Getty on line	On DG/UX systems, there should be a getty on a dial-in line but not on a dial-out line. For a bi-directional port, **uugetty** must be used.
Wrong line speed	Make sure the modem line speed (baud rate) is compatible with the entry in the **/etc/inittab** file.
Wrong permissions/access	The device file (such as ttyxx) should permit reading and writing for any user. Permissions should be set at 666. All files in **/usr/lib/uucp** and **/var/spool/uucp** should be owned by **uucp**.
Too many unsuccessful attempts	Remove the **STST** and lock (**LCK.**) files from the **.Status** directory and start **uucico** yourself to try and complete the job. You may also have to delete any temporary files (with prefix TM) 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	Check the modem and cable and make sure they are functioning properly. Check 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.
Bad pathname	Incorrect or illegal pathname.

NOTE: For more detailed, expert information on UUCP, see Appendix E in this manual.

UUCP Management Procedures

This section contains methods for administering UUCP with the **sysadm** program. Alternatively, you can manually edit the UUCP data files. For expert information, see Appendix E of this manual.

When you select **uucpmgmt** from the **sysadm** Main Menu, the following is displayed on your screen:

```
                   UUCP Management

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

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

Procedure 12.1: Add Entries to the Devices File

Purpose	To set up the devices that UUCP will use for file transfer connections.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**adddevice**
References	**uucp(1), cu(1)**

You add entries to the **Devices** file to specify which tty lines are to be used and how these lines are to be accessed. You must supply a tty number, a modem name, and a baud rate.

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

```
TTY number?  15 ↵

Local Modem Type? [hayes]  ↵

Modem Speed? [1200] 2400 ↵

Device entry for tty 15 has been added.

Do you want to create another device entry? [yes] n ↵

Press New Line to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.2: Delete Entries From the Devices File

Purpose	To remove devices no longer used by UUCP for file transfer connections.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**deldevice**
References	**inittab(4), uucp(1)**

Use this command to remove entries from **/usr/lib/uucp/Devices**. When you select **deldevice**, the system responds as follows:

```
TTY Number? 15 ↵

Device tty 15 has been deleted.

Delete another device? [yes] n ↵

Press New Line to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.3: Modify Entries in the Devices File

Purpose	To modify information on the devices in **/usr/lib/uucp/Devices**.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**moddevice**
References	**uucp(1)**

Use this command to make any changes to your devices in **/usr/lib/uucp/Devices**. Below, let's change the modem type and speed. When you select **moddevice**, the system responds as follows:

```
TTY number? 15 ↵

New TTY number? [15] ↵

Local Modem Type? [hayes] micom ↵

Modem Speed? [2400] 1200 ↵

Device entry for tty 15 has been modified.

Do you want to modify another device entry? [yes] n

Press New Line to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.4: List Entries in the Devices File

Purpose	To display all or part of the device entries in the **Devices** file.
Starting Conditions	administrative or multiuser state
sysadm menu	uucpmgmt
Commands	**lsdevice**
References	**Appendix E**

To display all devices, press New Line for the default. When you select **lsdevice**, the system responds as follows:

```
      TTY Numbers [all] <NL>

  Modem     Type     TTY     Dial TTY     Speed     Parameters
  -----     ----     ---     --------     -----     ----------
  hayes      ACU      15        --         1200         \D
  direct     direct   11        --         9600         \D
  penril     ACU      04        17         1200         \D

      Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.5: Add Entries to the Systems File

Purpose	To add the name of a remote computer system to **/usr/lib/uucp/Systems**. To add **uugetty** entries to **/etc/inittab**. To edit **/usr/lib/uucp/Systems** to add a "send carriage return" line.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**addsystem**
Note	If you add a system running **uugetty**, be sure to follow the steps for editing **/etc/inittab** and **/usr/lib/uucp/Systems** at the conclusion of this procedure.
References	**uucp(1), uugetty(1M), Appendix E**

This command adds remote system names to your **Systems** file. All systems that you want to connect with must be listed in this file. The **addsystem** command also queries for the name of the remote modem, the tty line, the baud rate, the terminal switch ID (if a terminal switch is being used), a login name for the remote system, a password, and a phone number.

If the remote system is connected via a direct line, then you should answer the "Remote Modem Type?" query with **direct**. In addition, for direct line connections, **addsystem** makes an entry in the **Devices** file in which the word "direct" is treated like a device name. If you want bidirectional communication on direct line connections running HoneyDanber UUCP, **uugetty** must be running at both ends.

Let's use **addsystem** for two cases: a direct line connection and a modem connection. Either of these cases may use a terminal switch. A terminal switch is an entry point at which incoming login requests must specify what system is desired. The terminal switch makes the initial connection, then the incoming requestor (daemon or human) must enter a login name and password.

Direct Connects

Below, let's assume that we've spoken with the administrator of a remote system named **dogfish8** and we've obtained all the information we need to add this system to the **Systems** file. We will be running **uugetty**.

When you select the **addsystem** command, the system responds as follows:

```
System Name?  dogfish8 ↵
Remote Modem Type? [hayes] direct↵

TTY Number? 21 ↵
Terminal Switch Machine ID? dogfish8 ↵

Login Name? uucp ↵
Password? fireball ↵

System entry for dogfish8 has been added.

Do you want to create another system entry? [yes] n ↵
Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Dial-up Connections

Let's use the same system, **dogfish8**, only this time, we'll assume that the connection will be via modem.

```
System Name?  dogfish8 ↵

Remote Modem Type? [hayes]  ↵

Modem Speed? [1200]  ↵
Phone Number? 444-5555 ↵

Terminal Switch Machine ID? dogfish8 ↵
Login Name? uucp ↵
Password? fireball ↵

System entry for dogfish8 has been added.

Do you want to create another system entry? [yes] n ↵
Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Editing the /usr/lib/uucp/Systems File

You need to add a line to any entry in your **Systems** file that will be running **uugetty**. This sends carriage returns to systems running **uugetty**. Below, we'll edit the modem example entry we just created in **Systems**:

Before:

```
dogfish8  Any ACU 1200 in:--in uucp ssword: fireball
```

After:

```
dogfish8  Any ACU 1200 "" \r\d\r\d\r\d\r in:--in: uucp ssword:fireball
```

The above script is needed because **uugetty** expects to read a character before outputting the login message.

Adding uugetty to inittab

If you want bidirectional communications with new UUCP (direct or dial-up), you must add **/usr/lib/uugetty** and some options to the *process* field of any tty line you want to use for UUCP communications. The **sysadm adddevice** command will create tty entries in **inittab**. You need to make the following changes with an editor: change `/etc/getty` to `/usr/lib/uugetty -r -t 60`.

```
tt15:23:respawn:/usr/lib/uugetty -r -t 60 tty15 1200
```

The **-r** option causes **uugetty** to wait before sending out the login message. The **-t 60** option sets the timeout delay to sixty seconds.

Connecting Like and Unlike Versions of UUCP

This section shows some example connection setups for various versions of UUCP. Use these, or configure your own connections.

New UUCP to Old UUCP: Direct or Dial-up Connection

In **/etc/inittab** ---

- If New uses **off** and **getty**, then Old uses **respawn** and **getty**.

  There is no bidirectional communication here. The computer with the tty line turned **off** will have to poll the other computer. In this example, new UUCP would have to call old UUCP. Old UUCP cannot call New UUCP.

New UUCP to New UUCP: Direct Connection

In **/etc/inittab** ---

- Both computers *must* have **respawn** and **uugetty** in order to have bidirectional communication. With anything else in **inittab**, communications will not work.

New UUCP to New UUCP: Dial-up Connection

In **/etc/inittab** ---

- Both could have **respawn** and **uugetty**. Both would then have bidirectional capability.
- One could have **respawn** and **uugetty**, the other could have **off** and **uugetty**. The one set to **off** might choose this purposely not to have bidirectional capability. That is, the administrator doesn't want to allow other machines to call in.

Procedure 12.6: Delete Entries in the Systems File

Purpose	To remove remote system names from the **Systems** file.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**delsystem**
References	**inittab(4), uucp(1)**

When you stop UUCP communications with a remote system, use this command to remove that system from **/usr/lib/uucp/Systems**. When you select **delsystem**, the system responds as follows:

```
System Name? mongo5 ↵

System mongo5 had been deleted.

Delete another system? [yes] n ↵

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.7: Modify Entries in the Systems File

Purpose	To change all or parts of the entries of a given **Systems** file entry.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**modsystem**
References	**inittab(4), uucp(1)**

Below, let's assume that the modem type and phone number have changed at the remote system connected via modem named **dogfish8**.

When you select the **modsystem** command, the system responds as follows:

```
System Name?  dogfish8 ↵

New System Name? [dogfish8]  ↵

Remote Modem Type? [hayes] att400 ↵

Modem Speed? [1200]  ↵

Phone Number? [444-5555] 444-4444 ↵

Terminal Switch Machine ID? [dogfish8]  ↵

Login Name? [uucp] ↵

Password? [fireball] ↵

System entry for dogfish8 has been modified.

Do you want to modify another system entry? [yes] n ↵

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.8: List Entries in the Systems File

Purpose	To display remote system names in the **Systems** file.
Starting Conditions	administrative or multiuser state
sysadm menu	uucpmgmt
Commands	**lssystem**

To list all of the computer systems that you can communicate with via UUCP, use the **lssystem** command. The system responds as follows:

```
        System Names? [all] <NL>

  Name      Modem     Speed     Phone/Token       Login String
 -------    -----     -----     -----------       ------------
 dogfish8   hayes     1200       444-4444         uucp/fireball
 opus       micom     1200       444-0004         uucp/delray


        Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.9: Add Entries to the Poll File

Purpose	Add times for polling remote systems to the **/usr/lib/uucp/Poll**.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**addpoll**
References	**uucp(1), Appendix E**

Systems listed in **Poll** can be polled at specified times. This means that at a given hour, a connection is made and any waiting files are transferred. A poll entry consists of the name of the system to be called and the hours when calls are to be attempted.

Below, let's set the polling times for remote system **dogfish8**. Set hours using the numbers 0 to 24. Enter **all** to poll on every hour. When you select the **addpoll** command, the system responds as follows:

```
Polled System Name? dogfish8 ↵

Polling Hours?  2  6  10  14↵

Do you want to create another poll entry? [yes] n ↵

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.10: Delete Entries in the Poll File

Purpose	To remove remote system names from **/usr/lib/uucp/Poll**.
Starting Conditions	administrative state
sysadm menu	uucpmgmt
Commands	**delpoll**
References	**uucp(1), Appendix E**

When you select the **delpoll** command, the system responds as follows:

```
Polled System Name?  opus ↵

Poll entry for opus has been deleted.

Delete another poll entry? [yes] n ↵

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.11: Modify Entries in the Poll File

Purpose	To change polling times to remote systems.
Starting Conditions	administrative or multiuser state
sysadm menu	uucpmgmt
Commands	**modpoll**
References	**uucp(1), Appendix E**

Use this command to change the times that remotes systems are called. When you select the **modpoll** command, the system responds as follows:

```
Polled System Name? dogfish8 ↵

New Polled System Name? [dogfish8] ↵


Polling Hours? [2  6  10  14] 1 4 8↵

Do you want to modify another poll entry? [yes] n ↵

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.12: List Entries in the Poll File

Purpose	To display remote systems and their polling times.
Starting Conditions	administrative or multiuser state
sysadm menu	uucpmgmt
Commands	**lspoll**

When you select the **lspoll** command, the system responds as follows:

```
Polled System Names? [all] <NL>

  Name          Polling Hours
---------      -------------
dogfish8          1  4  8
guss              2  5

Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Procedure 12.13: Test a UUCP Connection

Purpose	To determine if a UUCP connection has succeeded or failed.
Starting Conditions	administrative or multiuser state
sysadm menu	uucpmgmt
Commands	**trysystem**
References	**Uutry(1M), cu(1), uucp(1)**

After you have setup UUCP to communicate with a remote system, you can test that connection with this command. The **trysystem** command invokes the **uucico** daemon to start a connection with the system you specify. The **uucico** daemon prints information on the attempted connection. That output will look similar to the following, an example of a failed connection. When you select the **trysystem** command, the system responds as follows:

```
System Name? dogfish8 ↵

./uucico -r1 -sdogfish8 >/tmp/dogfish8  2>&1&
name (dogfish8) not found; return FAIL
_Request (FALSE), _Switch (TRUE), _CallBack (FALSE),
chdir(/var/spool/uucp/dogfish8)

Device Type dogfish8 wanted

Requested Device Type Not Found
Call Failed: NO DEVICES AVAILABLE
Conversation Complete: Status FAILED

Try another system? n ↵
Press the New Line key to see the uucpmgmt menu [?, ^, q]:
```

Output from **trysystem** will vary depending on the situation. Above, we see that device type **dogfish8** was wanted, but not found, so the call failed. Finally, the status of the connection attempt is given as FAILED. To fix this problem, first check your **Devices** file and make sure that the line is correct for **dogfish8**. If you have added device entries with **adddevice**, then your entries should be correct. But if you have edited the file manually, you may have introduced an error. If all information is correctly entered, then you may want to verify information with the administrator of **dogfish8**.

End of Chapter

Chapter 13 Network Management

Information in this chapter is useful only if your system is running TCP/IP, NFS, or the Yellow Pages.

This chapter shows you how to use **sysadm** to add, delete, and modify hosts, networks, and ethernet addresses after you have already installed your network software. We do not show you how to install, maintain, or use TCP/IP, NFS, or the Yellow Pages in this chapter. For such information, see

- *Installing and Managing DG TCP/IP (DG/UX™)*
- *DG TCP/IP (DG/UX™) User's Manual*
- *Managing NFS and Its Facilities on the DG/UX™ System*

Network Management Terms

We use the following terms in this chapter:

hostname This is normally the name of a remote system; it can, however, be a local system. The hostname is assigned by that system's administrator. The first character must be a letter. The remaining characters can be:

a-z A-Z 0-9 . (period) – (dash)

Names cannot end with a period or dash. Maximum name length is 255 characters.

host address The host address is actually the Internet address of a host with which you wish to communicate. This address is composed of four fields separated by periods. Each field contains a decimal number which represents a byte value.

```
000.000.000.000
```

The first field of the Internet address is in the range 1-223, inclusive. The remaining fields are in the range 0-255. This address is divided into a *network* part and a *host* part depending

	on the address class. The address class (A, B, or C) is determined by the number in the first field, i.e, the network part. Remaining fields are arbitrary specifications by the host, i.e., the host part.
Class A address	A class A address has a first number in the range 0-127, inclusive. Only the first number (90 below) describes the network address. **90**.1.2.3
Class B address	A class B address's first-field number is in the range 128-191, so that the network address part is composed of the first and second fields (128.1 below). **128.1**.2.3
Class C address	A class C address's first-field number is in the range 192-223, so that the network address part is composed of the first, second, and third fields (193.1.2 below). **193.1.2**.3

/etc/hosts	A database containing Internet addresses and hostnames in the following format: `internet_address hostname`.
	
	`129.222.1.86    sys86`
	`115.1.2.10      sys10`

/etc/networks	Contains network names and network addresses in the following format: `network_name  network address`. `fishnet   129.222` The entry `fishnet` is arbitrarily chosen to specify the network address 129.222 (class B). This alias is useful to network programs such as **route(1M)**. The alias may also be used in system startup scripts (such as an **rc** script).
/etc/tcpip.params	Contains parameters for various commands (**hostname(1)**, **hostid(1)**, **security(1M)**, **route(1M)**, and **ifconfig(1M)**) invoked by the **rc** scripts to initialize the network. These parameters are set for one interface when you use the **sysadm installpkg** command. If you add another board or want to update existing entries, you will have to change the parameters listed in the **tcpip.params(4)** manual page.

/etc/nfs.params	Contains parameters used by **rc** scripts in **/etc/init.d** to initialize the Yellow Pages data base.
YP master	Yellow Pages master. The single machine in the network that holds the master networks and hosts files that are exported to other machines. Global changes can be made only on the YP master machine.

Network Management Procedures

When you select **networkmgmt** from the **sysadm** Main Menu, the following menu is displayed:

```
                  Network Management

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

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

You will do most of your network operations with host selections 1, 2, 3, and 4 from the Network Management menu. These selections allow you to make network changes at the same time as you are making host changes. You will probably use the remaining sections less frequently, but they are provided for your convenience.

NOTE: The **sysadm** program checks to see if you have NFS loaded. If you do not, your system will not display any of the YP queries that are in the **networkmgmt** procedures.

Procedure 13.1: Add Hosts

Purpose	To add host names and addresses to **/etc/hosts**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**addhost**
References	**hosts(4), gethostent(3N)**

The **addhost** command takes a host name and address and adds it to **/etc/hosts**. If the host is the YP master, **addhost** queries you to select either the global or local file for modification. If the network portion of the address is not in **/etc/networks**, then **addhost** requests a unique network name and adds that network name to **/etc/networks**.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

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

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

Next, you will be queried for the name of the new host you wish to add and for a host Internet address. Let's assume that name is **sys86** with class B address 129.222.1.46:

```
Host name?  sys86 ↵
Host address? 129.222.1.46 ↵
YP Server? [yes] ↵
```

We took the default for the YP server question. Enter **n** if the host is a YP client. If the network portion of the host address (129.222) is not in **/etc/networks** file, the following is displayed:

```
The network address 129.222 does not currently have a name.
Please give it one.

Network name? fishnet ⮐

The entry for sys86 has been added.
Do you want to add another host? [no]
```

You can loop back through this procedure and add another host if you wish by typing **y** to the last query. But if you are finished and you are the YP master, the system displays

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

Procedure 13.2: Delete Hosts

Purpose	To delete a host name from **/etc/hosts**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**delhost**
References	**hosts(4), gethostent(3N)**

Delhost requests the host name; if the host name exists, it is deleted from **/etc/hosts**.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

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

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

You may use **lshost** to see the names of existing hosts.

```
Host name?  sys00 ↵

The entry for sys00 has been deleted.

Do you want to delete another entry? [no]
```

If you are the YP master, when you are finished, the system displays:

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

Procedure 13.3: Modify a Host Entry

Purpose	To change the name and/or address of a host.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**modhost**
References	**hosts(4), gethostent(3N)**

If the local host is the YP master, **modhost** asks you to select the global or local file for modification. Then **modhost** requests the current host name. If the name does not exist, **modhost** asks again, then you may change either the host name or the address. In each case, the new entry must be either the same as the old or unique. The file **/etc/networks** is not changed.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files. When you select **modhost**, the system responds as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.

Host name?  sys86 ↵
New host name?  sys00 ↵
Host address? [129.222.1.46] ↵
YP Server? [yes] ↵
The entry for sys86 has been modified.
Do you wish to modify another host? [no] ↵
```

Above, we changed **sys86** to **sys00**. We hit New Line and kept the old (default) host Internet address. If you are the YP master and you are finished modifying entries, the system displays

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

Procedure 13.4: List Hosts

Purpose	To display the contents of **/etc/hosts**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**lshost**
References	**hosts(4), gethostent(3N)**

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

When you select **lshost**, the system respond as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

Assuming you are not the YP master, your system displays

```
Host Names(s)? [all] ↵

Host            Address
------          ---------
localhost       127.1
sys86           129.222.1.46
sys10           129.222.1.10
```

Procedure 13.5: Add Networks

Purpose	To add a network name and address to **/etc/networks**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**addnetwork**
References	**networks(4)**

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

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

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

Next, you will be queried for the name of the new network to be added. This name must not already exist in **/etc/networks**.

```
Network name?  newnet ↵

Network Address? 192.114.25 ↵

The entry for newnet has been added.
Do you want to add another network? [no] ↵
```

If you are the YP master, when you are finished, the system displays

```
Updating the YP Yellow Pages password and group maps.
```

Procedure 13.6: Delete Network Names

Purpose	To delete a network name from **/etc/networks**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**delnetwork**
References	**networks(4)**

If the host is the YP master, **delnetwork** asks you to select either the global or local file for modification. **Delnetwork** requests the host name; if the host name exists, it is deleted from **/etc/networks**.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

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

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

You may use **lsnetwork** to see the names of existing network names.

```
Network name?  newnet-24 ↵

The entry for newnet-24 has been deleted.

Do you wish to delete another entry? [no]
```

If you are the YP master, when you are finished, the system displays:

```
Updating YP Yellow Pages password and group maps.
```

Procedure 13.7: Modify Networks

Purpose	To change the name and/or address of a network.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**modnetwork**
References	**networks(4)**

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files. When you select **modnetwork**, the system responds as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

Next, you will be queried for the name of the network to be modified. This name must already exist in **/etc/networks**.

```
Network name?  newnet ↵
New Network name? [newnet]  newnet-24 ↵

Network Address? [192.114.25] ↵

The entry for newnet has been modified.
Do you want to modify another network? [no]
```

Above, the first query locates the network you wish to modify and uses the information to supply the default fields of the next two queries. Notice that we changed **newnet** to **newnet-24**. We left the network address as it was by hitting the New Line key. If you are the YP master, when you are finished modifying a network name, the system displays:

```
Updating the YP Yellow Pages password and group maps.
```

Procedure 13.8: List Networks

Purpose	To display the contents of **/etc/networks**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**lsnetwork**
References	**networks(4)**

If you are the YP master, **lsnetwork** asks you to select either the global or local file for listing. The entries from the hosts file are listed.

```
Access the Global Host/Network List? [yes] ↩

Network Name(s)? [all] ↩


Network          Address
-------         --------
fishnet         129.222
butterflynet    89.3
satnet          128.5
newnet-24       192.114.25
```

Procedure 13.9: Add an Entry to /etc/ethers

Purpose	To add an entry to the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**addether**
References	**ethers(4)**, *Installing and Managing DG TCP/IP (DG/UX™)*

A server machine that will be providing operating system service to a diskless client needs the client's ethernet address so that the client can boot over the network. The netboot sequence is explained in Chapter 1. The **addether** function maps ethernet addresses in hex notation to client host names. Let's assume that we are adding a diskless client whose ethernet address is 2:0:20:0:J3:1 and whose host name is **dg2**. We begin by typing:

```
# sysadm addether ↵
```

The system responds as follows:

```
Host Name? dg2 ↵
Ethernet Address? 2:0:20:0:J3:1↵
The entry for dg2 has been added.
Do you want to add another entry? [yes] no↵
```

Procedure 13.10: Delete an Entry from /etc/ethers

Purpose	To delete an entry from the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**delether**
References	**ethers(4)**, *Installing and Managing DG TCP/IP (DG/UX™)*

This function removes entries from the file **/etc/ethers**. Let's remove the entry for the diskless client **dg2**. We begin by typing:

```
# sysadm delether ⤸
```

The system responds as follows:

```
Host Name? dg2 ⤸
The entry for dg2 has been deleted.
Do you want to delete another entry? [yes] no ⤸
```

Procedure 13.11: Modify an Entry in /etc/ethers

Purpose	To modify an entry in the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**modether**
References	**ethers(4)**, *Installing and Managing DG TCP/IP (DG/UX™)*

If you need to change entries in the **/etc/ethers** file, use the **modether** function. As an example, we'll modify the entry for diskless client **dg2**. We begin by typing:

```
# sysadm modether ↵
```

The system responds as follows:

```
Host Name? g2 ↵

Ethernet Address? [2:0:20:0:J3:1] 3:0:20:3:J3:3 ↵

Do you want to modify another ethers entry? [yes]
```

Procedure 13.12: List Entries in /etc/ethers

Purpose	To display the entries in the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**lsether**
References	**ethers(4)**, *Installing and Managing DG TCP/IP (DG/UX™)*

If you need to to know the ethernet address of the hosts on your servnet, use the **lsether** function. This function produces the following:

```
 Host             Ethernet Address
------            ----------------
dg1               2:0:20:0:D3:1
dg2               2:0:20:0:J3:1
```

Procedure 13.13: Set NFS and YP Parameters

Purpose	To edit the parameters in **/etc/nfs.params**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**nfsparams**
References	**nfs.params(4)**, *Using the Network File System on Your DG/UX™ System*

The **nfsparams** function edits the **/etc/nfs.params** file. This file defines the configuration parameters for the Network File System (NFS) and the Yellow Pages (YP). The **nfsparams** command only works on the domain name and the host type. You must make other changes manually.

The **nfsparams** function displays the current values as defaults. We use `accounts` below as an example domain name.

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

```
Domain Name? [accounts] ↵

Yellow Pages Host Type? [server] ↵

Do you want to use these settings? [yes] ↵
```

We selected the defaults above. Simply type in any changes you want at each query.

Procedure 13.14: Set TCP/IP Parameters

Purpose	To edit the parameters in **/etc/tcpip.params**.
Starting Conditions	administrative or multiuser mode
sysadm menu	networkmgmt
Commands	**tcpipparams**
References	**tcpip.params(4)**, *DG TCP/IP (DG/UX™) User's Manual*

The **tcpipparams** function allows you to change the hostname and hostid parameters in the file **/etc/tcpip.params**. This file defines the TCP/IP parameters needed to configure the network at boot time. In addition, this function queries you to include or exclude each interface in your list of network interfaces (hken0,inen0, etc.). To include an interface means that it will be part of the configuration. When you include an interface, you will be asked to define a hostname, a broadcast address, and a netmask for the interface. Below, let's assume that there are two entries in your **system.tcpip** file: a network board (hken) and a pseudo-device (loopback). Both of these must already have an entry in **/etc/hosts**; those entries in this case could look like:

```
129.222.1.86    sys86
127.0.0.0       localhost
```

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

```
Local Host Name? sys86 ↲
```

Next, **tcpipparams** will go through all the interfaces (network boards) you have listed in your **/usr/src/uts/aviion/cf/system.tcpip** file. These entries correspond to device files in **/dev**. You will be asked if you want each interface to be configured at boot time. We'll show one interface below for hken0 which we are assuming is the first device entry in **system.tcpip**. Note that loopback does not use the netmask or the Broadcast Address, so we'll just take the defaults.

```
Configure hken0? [yes] ↲
Host Name for hken0? sys86 ↲

Netmask for hken0? [0Xffff0000] ↲
Use 1-bits in Broadcast Address for hken0 [yes] no ↲

Configure loop0? [yes] ↲
Hostname for loop0?  localhost ↲
```

```
Netmask for loop0? [0Xffff0000] ↵
Use 1-bits in Broadcast Address for loop0 [yes] ↵

Do you want to use these settings? [yes] ↵
These settings have been placed in /etc/tcpip.params.
They will take effect the next time you boot DG/UX.
```

If there are other device entries, the function will start over.

End of Chapter

Chapter 14
User Account Management

This chapter is for systems supporting one or more user accounts. As a timeshare system manager, you will set up and manage the everyday working world in which users will function. This task includes adding and removing user accounts, creating aliases and groups, maintaining system security, answering users' questions, and helping users with system problems. If you are part of a YP data base, these tasks will be done via the master server host.

The major sections of this chapter are:

- User Account Terms
- About User Accounts
- Security Suggestions
- User Account Procedures
- The User's Environment
- Expert User Management Information

User Account Terms

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

login name	A valid name for logging on to the system. Also known as *username*. A login name can be up to eight alphabetic or numeric characters; the first character must be alphabetic.
password	A unique string that allows a user access to the system. A password must be at least six characters, and a maximum of eight characters. At least one character *must* be a numeral or a special character. You may set explicit passwords or leave a password field open when creating a user account with **adduser**. If the field is open, then users are prompted to set a password when they first login.

password aging	A system which forces users to set new passwords within a specified number of weeks. An "aged" password is one that must be changed within the specified number of weeks. When this time period lapses, the password will no longer gain a user entry to the system. The user must choose a new password. System administrators usually decide if they want to use password aging or not. See "Expert User Management Information" for details.
group	A set of users with access privileges to the same set of files based on group ID numbers. Also known as *groupname*. People that need access to the same files can be listed in a group. For example, anyone in group "prog" could access those files associated with that group name. Other people in, for instance, group "pool" would not have access to the files of group "prog".
alias	A mailing list. An alias contains one or more login usernames. If you address mail to an alias, the mail is delivered to all users listed in that alias. An alias entry consists of an alias name and a list of alias members.
user ID	A unique number that identifies a user to the system. The user ID number (**uid**) is between 100 and 60,000. The number must not include a comma. Superuser (root) uses 0. System ID numbers are 1 to 99. The **sysadm** program supplies new ID numbers by default.
group ID	A unique number that identifies a group to the system. The same conditions apply to the group ID (**gid**) number as to the **uid**. The **sysadm** program supplies new ID numbers by default. System ID numbers are 1 to 99.
home directory	The origination point of the user's directory tree. The home directory is where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory such as **/mach_2/poulet**.
initial program	The program invoked at the time the user logs in. Choices include **sh**, **csh**, or some other local executable program. Users typically select their own initial program.
YP master server	The YP master server is the single machine in the network that holds the master networks and hosts files that are exported to other machines. Global changes can be made only on the master machine. If your system has the Network File System (NFS), and the current machine is the YP master server, you will be asked to choose local or global options in queries for **usermgmt** commands in this chapter.

Request for User Login

You may find it helpful to use a standard form for people who wish to become users on your system. The following is a suggested form for collecting information on new users.

```
*********************************************************

                 REQUEST FOR USER LOGIN


1. Full name _________________________________________

2. Login name ________________________________________

3. Login group _______________________________________

4. Other groups ______________________________________

5. Login directory ___________________________________

6. Initial program (shell): sh  csh  other ___________

7. Mail aliases ______________________________________


*********************************************************
```

Figure 14-1 User Login Request Form

About User Accounts

Because of some preset defaults, you can begin adding users to the system immediately. Using **sysadm userdefaults**, you can change the defaults for passwords, groups, and initial programs. Use **modalias** to change alias defaults. The following list shows how defaults are originally set on the DG/UX system:

- Password aging -- Set to *off*.
- Default group -- Set to *general*.
- Default initial program -- Set to */bin/sh*.
- Default mail aliases -- Set to *everybody*.

- Default parent directory -- Not set. You *must* set this one.

When you are ready, read the section "The User's Environment." 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.

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 Procedure 14.10: Add Mail Aliases for details.

Parent Directory and Initial Program

We do not provide a default parent directory because we can't be sure of how you've laid out your logical disks and file systems when you installed the DG/UX system. You will have to indicate the name of the file system that you wish to make the default parent directory for users.

If you have or plan to have NFS in the future, then you should make sure that parent directories have unique names across machines. One way to do this is by using the hostname as the parent directory name. We did this with our example installation system in Chapter 2. We created a logical disk and file system named **mach_2** intended especially for user accounts. So, to ensure that NFS functions properly for Nick New-user, we would assign him to directory **/mach_2**, which is also the hostname of Nick's machine.

The default intial program is traditionally specified as **/bin/sh**, but it can be any executable local shell program.

Security Suggestions

The following are suggestions for maintaining a secure system:

- Set the access permissions to directories and files to allow only the necessary permissions for owner, group, and others.

- All logins should have passwords. Advise users to change passwords regularly. Advise users not to pick obvious passwords. Password aging is an option.

- All dial-up ports should have passwords. Any system with dial-up ports is not really secure.

- Users who make frequent use of the **su** command can compromise the security of your system by accessing files belonging to other users without the other users' knowledge. The more people who know a given login and password, the less secure access is to the system. For this reason, a log is kept on the use of the command. Check the file **/usr/adm/sulog** to monitor use of the **su** command.

- Login directories, **.profile** files, and files in **/bin**, **/usr/bin**, and **/etc** should not be writable by others.

- Encrypt sensitive data files. The **crypt(1)** command together with the encryption capabilities of the editors (**ed** and **vi**) provide protection for sensitive information.

- Do not leave a logged-in terminal unattended, especially if you are logged in as root.

User Account Procedures

The following procedures are covered in this section:

- Set user defaults
- Add, delete, modify, or list user accounts
- Add, delete, modify, or list groups
- Add, delete, modify, or list aliases

When you select **usermgmt** from the **sysadm** Main Menu, the following is displayed on your screen:

```
                  User Management

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

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

Procedure 14.1: Set User Defaults

Purpose	To set defaults for group name, parent directory, initial program, and password aging.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**userdefaults**
Note	The DG/UX system is shipped with many defaults already set. Check the entries in **userdefaults** to make sure they are defaults you want.
References	**group(4), sh(1), csh(1)**

This command sets the defaults that the **adduser** command uses. If you set defaults first, **adduser** is much easier to use. With the **userdefaults** command, you can set defaults for group names, parent directories for home directories, initial programs (shell), and password aging.

If you're running NFS, **sysadm** will determine if your machine is the YP master server. If it is, you will be asked if you want to access the global or local user lists. If your machine is not the YP master server, you can only access local user lists.

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

```
Enable Password Aging? [no]  y ↵
```

Enter **yes** if you want to place time limits on user passwords. Enter **no** if you do not want to use password aging. If you use password aging, you will be asked to define a minimum and maximum age in weeks for passwords. Users will be forced to change passwords when the maximum time expires. Users will not be allowed to change a password until the minimum time expires. If you use password aging, the limits you set will apply to all users added later. Above, we chose to use password aging by answering **yes**. Next, the system asks for a maximum and minimum age in weeks. These numbers must be between 0 and 63, inclusive.

```
Maximum Password Age? 8 ↵

Minimum Password Age? 1 ↵
```

After password aging, you can set group defaults:

```
Group Name? [general]  ↵

Parent directory of login directory? /mach_2 ↵

Initial Program? [/bin/sh]   ↵

Press the New Line key to see the usermgmt menu [?, ^, q]:
```

When you are satisfied with the defaults, you are ready to add users to the system.

Procedure 14.2: Add User Accounts

Purpose	To add one or more user accounts.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**adduser**
References	**passwd(4), passwd(1), group(4), newgrp(1)**

For each user account to be added, **sysadm** requests the user's full name, the login name, a user ID, a login group name, the parent directory of the user's home directory, and the initial program. As an example, let's add user L.Q. Poulet to our system. If you select the **adduser** command, the system responds as follows:

```
User Login Name? poulet ↵

Full User Name? L.Q. Poulet ↵
```

Next, you are queried for the user ID. This is a number the DG/UX system uses to associate files with a given login name. Each user has only one user ID. User ID 0 is reserved for the superuser. Numbers between 1 and 99 are reserved for other system logins, like **lp**. Regular user IDs are between 100 and 60,000. You may select a default ID number by responding with a New Line; this gives you the next unused ID number.

```
User ID? [101]  ↵
```

The next query asks for the group to which you want Poulet to belong. You can type the name of an existing group, or you can answer with the default, putting Poulet into group *general*. If you are just setting up users for the first time, you may wish to add them to the default group for now. Later, you can add users to specific groups as needed. For now, let's put user Poulet into the default group *general*:

```
Group Name? [general] ↵

Parent directory of login directory? [/mach_2] ↵
```

If you do not want Poulet to go into the default directory, enter a full pathname of the parent directory where Poulet's login directory will be created. The login directory has the same name as the user. For example, if you choose **/mach_2** as the parent directory, then Poulet's login directory will be **/mach_2/poulet**. You can override the default here, but if you type the name of a directory that does not exist, such as **/messages**, the system will respond:

```
Warning: The directory /messages does not exist.
Do you wish to create /messages?
```

You can create the new parent directory by answering **y** to the last query. But let's assume you selected the default parent directory. When user Poulet logs in, his home directory will be **/mach_2/poulet**.

Now that Poulet has a home directory, he needs to be assigned an initial shell program. Users will generally request a certain shell, usually **csh** or **sh**. We'll assume that user Poulet has asked for the **csh**. See **sh**(1) and **csh**(1) for details on both of these programs.

The system asks for the initial program that you want to assign to user Poulet. Below, we'll give Poulet the C shell. Poulet's local profile is also determined when you set his initial program. See "The User's Environment" later in this chapter for more information.

```
Initial Program? [/bin/sh] /bin/csh ↵
```

The next query determines whether or not Poulet will be able to log in with a password.

```
The password is currently clear.
Password Operation? [set] ↵
```

Responses to the Password Operation query are **set** or **disable**. With **set** you can type in a password for the new user, or you can simply hit New Line and no password is assigned. If you don't assign the user a password here, then the user will be instructed to enter a password at the first log on. Selecting **disable** means that this user can never login to the system.

The **adduser** command now gives you a chance to do one of the following with the entries for username Poulet:

```
Do you want to edit, skip, or install this user entry?
[install] ↵

User poulet has been added.
```

None of the entries in **adduser** are recorded until you **install** them. We **installed** this user entry by pressing New Line. The choices are:

edit	Use current values as default and start over.
skip	Discard last values entered and quit.

install Add this user to the system.

If you choose, you can add another user by answering **y** to the next query. If you're finished, type **n** or **no** and you will exit from **adduser**.

```
Do you want to add another user?[yes] n ↵

Press the New Line key to see the usermgmt menu [?, ^, q]:
```

Poulet now has a user account on your system. The first time Poulet logs on to the system, he will be prompted to set his password. **Sysadm** invokes the **passwd(1)** program. All Poulet has to do is type in a password of his choice, say **rover8**. Now, every time Poulet logs on, he will not be allowed access to the system unless he types **rover8**.

Procedure 14.3: Delete User Accounts

Purpose	To delete one or more entries from the user account list.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**deluser**
References	**passwd(1), group(4)**

This command removes one or more user accounts from the system. When a user account is removed, the home directory, the mailbox, group entries, and mail alias entries are deleted for that user. To delete L.Q. Poulet's account, select the **deluser** command. When asked for the user name, give the login name.

```
User Login Name?  poulet
Do you really want to delete poulet? [no] y

User poulet 101 has been deleted.
Delete the Home Directory (/mach_2)? [no]  ⤶

Do you want to delete another user? [yes] ⤶
```

Above, we retained Poulet's home directory by pressing New Line for the default. We did this in case other users need access to this directory. So, in the above case, the user is deleted, but his directory remains. If you run the **ls -al** command on this directory, you will no longer see Poulet printed as owner; Poulet's ID number 101 will be listed instead. Delete this directory by hand when you are finished with it.

Remember, all group entries for login name poulet have also been deleted.

Procedure 14.4: Modify User Accounts

Purpose	To modify user account entries.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**moduser**
References	**passwd(4)**

This command uses the same queries as the **adduser** selection. Default values are whatever entries were made with **adduser** or subsequently modified with **moduser**. Let's say we wanted to modify some of the entries for user Poulet. We select the **moduser** command and the system responds as follows:

```
User Login Name? poulet ↵
New Login Name? [poulet] ↵
Full User Name? [L. Q. Poulet]  ↵


User ID? [101] ↵

Group Name? [general] ↵
Parent directory of login directory? [/mach_2] /mach_2/test ↵
```

We changed username Poulet's login directory from **/mach_2** to **/mach_2/test**.

```
Initial program? [/bin/csh] /bin/sh ↵
```

We changed Poulet's initial login program from **csh** to **sh**. Next, we can modify a user's password operation:

```
Password Operation? [keep]  ↵
```

We took the default, which "keeps" the user's password operation exactly as it was. We could have selected **disable** to prevent this user from logging on if we wanted, or we could have used set to change or delete the password.

```
Do you want to edit, skip, or install this entry? [install] ↵
Do you want to modify another user entry? [yes] n ↵
```

So, for Poulet, we changed his home directory and his initial program. Everything else remains the same.

Procedure 14.5: Display User Information

Purpose	To list user account entries.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**lsuser**
References	**who(1)**

This command prints information about all user accounts on the system. You can request information on all or specific users. The **lsuser** command will print the login names, user IDs, group names, login directories, and shells. Let's try both methods. First, we'll list all users, then we'll list information on one user, Dupree. Select the **lsuser** command. The system responds as follows:

```
User Name(s)? [all]  ↵

Login Name     UID    Group      Home Directory    Program
----------     ---    -----     --------------     -------
dupree         102    general  /mach_2/dupree       csh
kermit         103    general  /mach_2/kermit       sh
bond           104    largo    /mach_2/bond         csh


   Press the New Line key to see the usermgmt menu [?, ^, q]:
```

Next, let's get information on just one user, Dupree:

```
User Name(s)? [all] dupree

Login name:          dupree
Real Name:           Morton El Dupree
User ID:             102
Group:               general
Login Directory:     /mach_2/dupree
Initial Program:     csh
Password:            clear

Press the New Line key to see the usermgmt menu [?, ^, q]:
```

Procedure 14.6: Add Groups

Purpose	To add a new group name. To add group members.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**addgroup**
References	**group(4), newgrp(1)**

This command adds entries to the list of groups in **/etc/group**. A group entry consists of a group name, group number, and a list of group members. A group member is an established login name. In this procedure, let's create an example group named *writers*. Select **addgroup** from the **usermgmt** menu and the system responds as follows:

```
Group name? writers ↵
```

A group name can be up to eight characters long. Use upper or lowercase letters, numbers, or a combination of letters and numbers. Always use a letter for the first character. If you type a name that is already in use, the system responds:

```
Group name writers already exists.  If you wish to modify
this group, quit now and use modgroup.  Otherwise, choose
another name.
```

Next, you are queried for the group ID number. The **sysadm** program will issue ID numbers by default. However, you may override **sysadm** and enter ID numbers. If you do, choose a number between 101 and 60,000 that is not being used by any other group. If you choose a number outside the specified range, or a number already in use, **sysadm** will ask you to change your entry. Below, we'll take the default ID number by hitting New Line:

```
Group ID? [106]  ↵
The member list for writers is now empty.
```

Since *writers* has just been created, it has no members. The next query is:

```
Group Member List Operation? [list] end ↵
```

The default here lists members, but since there are no members in *writers*, we chose to type **end** to install the group name. Your choices for the "List Operation" query are:

add	Enter member names.
delete	Delete member names.
list	Display list of member names.
end	Writes entries to **/etc/group**.

To use any of the above operations, the member must have already been added with the **adduser** command. Invalid names will be ignored. Next, the system displays:

```
Group writers has been added.
Do you want to create another group? [yes] n ↵

Press the New Line key to see the usermgmt menu[?, ^, q]:
```

So, we have created an example group *writers*.

Procedure 14.7: Delete Groups

Purpose	To delete a group.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**delgroup**
References	**group(4), newgrp(1), grpck(1)**

This command deletes group entries from a group list. To list all groups, type **?** when you are asked for a group name as follows:

```
Group name?  ? ↵

Enter one or more group names to be deleted.
The list of available groups is:

general       101
farmers       102
ranchers      103
cowdogs       104
stooges       105
writers       106

Group name?  ranchers

Do you really want to delete ranchers 103? [no]  y
Group ranchers 103 has been deleted.

Do you want to delete another group? [yes] n ↵
Press the New Line key to see the usermgmt menu[?, ^, q]:
```

All files associated with group **ranchers** will now be listed by group ID number instead of by group name. If a user tries to change groups (with the **newgrp(1)** command), that user will receive an error message stating that the group, in this case **ranchers**, does not exist.

Procedure 14.8: Modify Groups

Purpose	To modify entries in the group list.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**modgroup**
References	**group(4), newgrp(1), grpck(1)**

This command modifies entries in the list of groups. You can change a group name, a group ID, or a group member. To modify a member list, you will use the add, delete, and list operations. After modifying one group, you have the option of modifying another or exiting **modgroup**. Below, let's modify the group *cowboys*. If you select the **modgroup** command, the first query is:

```
Group Name?  cowboys ↵
New Group Name? [cowboys] cowdogs ↵
Group ID? [104] ↵

Group Member List Operation? [list] delete ↵
User Name(s) to be deleted? roy ↵

User Name(s) roy has been deleted.
```

After deleting, let's check to see if the deletion has occurred.

```
Group Member List Operation? [list] list ↵

Group Name  GID      Members
----------  ---   -------------------
cowdogs     104   paladin  cisco  poncho  gene


Group Member List Operation?  end ↵

Group cowboys has been modified.
Do you want to modify another group? [yes] n↵
Press the New Line key to see the usermgmt menu[?, ^, q]:
```

As the display shows, member *roy* has been deleted.

Procedure 14.9: List Groups

Purpose	To print group entries.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**lsgroup**
References	**group(4), grpck(1)**

This command prints one or more group entries. A group entry consists of a group name, a group ID, and group members. Group members are listed by their login names. You can list all group entries by hitting New Line for the default value in the first query. If you select **lsgroup**, the system displays:

```
Group name(s)? [all] ⮐

Group Name    GID       Members
----------    ---    ----------------------
 general      101    (all logins)
 farmers      102    grandpa, luke, pepino
 cowdogs      104    paladin, cisco, poncho, gene
 stooges      105    moe, larry, curly, shep
 writers      106    woody, bugs, daffy, elmer

Press the New Line key to see the usermgmt menu[?, ^, q]:
```

Procedure 14.10: Add Mail Aliases

Purpose	To add an alias to the list of mail aliases.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**addalias**
References	**group(4)**

A mail alias is a mailing list. An alias contains one or more login user names. If you address mail to an *alias*, you can send mail to all users listed *in* that alias. The **addalias** command adds names to alias entries. An alias entry usually consists of an alias name and a list of alias members. But, for now, let's make an empty mail alias called *everybody*; this will be the default alias that the **userdefaults** command will read when you add users to the system. When you select the **addalias** command, the system responds as follows:

```
Alias Name?  everybody ↵
```

An alias name can be from 1 to 32 characters long. Use upper or lowercase letters, numbers, or a combination of letters and numbers. Always use a letter for the first character. The next query gives us the choice of using add, delete, list, or end as operations to perform on an alias. Below, let's just install the alias *everybody*. Later, if you choose, you can use the *add* operation to put members in this alias. The system displays:

```
Alias Member List Operation?  end ↵

Do you want to create another alias? [yes] n ↵
Press the New Line key to see the usermgmt menu[?, ^, q]:
```

By typing **end**, we added the empty alias *everybody* to the file **/usr/lib/aliases**. Remember to type ? if you need additional information on aliases.

Procedure 14.11: Delete Mail Aliases

Purpose	To delete an alias from the mail alias list.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**delalias**
References	**mailx(1)**

This command deletes one or more alias entries from the mail alias list. When deleted, the alias is no longer available to the mail system. Let's say that earlier you used the **addalias** command to create alias *pawns*. Now you want to delete *pawns*. If you select **delalias**, the system responds as follows:

```
Alias Name? pawns ↵
Do you really want to delete pawns? [no] y ↵

Alias Name pawns has been deleted.

Do you want to delete another alias? [no] ↵

Press the New Line key to see the usermgmt menu[?, ^, q]:
```

Procedure 14.12: Modify Mail Aliases

Purpose	To modify entries in the list of mail aliases.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**modalias**
References	**mailx(1), sendmail(1C)**

This command requests an existing alias name. If the name is valid, you can change the name and member list. You will use the add, delete, list, and end operations. Below, let's delete and add a member in alias **pawns**. If you select the **modalias** command, the system responds as follows:

```
Alias Name?  pawns
New Alias Name? [pawns] ↵

Alias Member List Operation? [list] delete ↵
Delete Mail Address(es)?  kermit ↵

Member Name(s) kermit has been deleted.

Alias Member List Operation? add ↵
Add Mail Address(es)? lucy@sys5 ↵

Member Name(s) lucy@sys5 has been added.

Alias Member List Operation? end ↵

Do you want to modify another alias? [no]  ↵

Press the New Line key to see the usermgmt menu[?, ^, q]:
```

Above, we deleted *kermit* and added *lucy@sys5*. We typed **end** when we were ready to record these changes to the list. Finally, we answered **n** to the last query to exit **modalias**.

Procedure 14.13: List Mail Aliases

Purpose	To print the list of mail alias entries.
Starting Conditions	administrative or multiuser
sysadm menu	usermgmt
Command	**lsalias**
References	**sendmail(1C)**

This command prints one or more alias entries. An alias entry consists of an alias name and alias members. The list of alias members includes mail login names for local aliases and addresses for remote aliases.

If you select the **lsalias** command, the system responds as follows:

```
Alias Name(s)? [all]   ↵

Alias Name          Members
----------          --------------------
everybody           luke, grandpa, pepino, dupree,
                    kermit, bond, paladin, cisco@sys3,
                    poncho, gene, moe, larry, curly,
                    shep@sys5, lucy@sys5

pawns               kermit, pepino, curly

Press the New Line key to see the usermgmt menu[?, ^, q]:
```

The User's Environment

The profile is the key element in establishing an environment in which users can successfully communicate with the computer.

Among the things that a profile can contain are:

- A PATH (searchlist) specifying files to be read by the system.
- Definitions of TERM (terminal) and TZ (time zone) variables.
- A command that prints **/etc/motd** upon login.
- Instructions to print notification of mail.

Profiles are of two types: global and local.

1) **The Global Profile**

   This is an ASCII text file, **/etc/profile** for **sh** users or **/etc/login.csh** for **csh** users, that can contain commands, shell procedures, and environment variables. Whenever a user logs in, the **login** process executes the global profile. The global profile allows people to immediately begin using the system after an account is set up. Later, they can modify their environments with the individual (local) profile. A sample global profile follows:

```
# /etc/profile:  customize this file as needed.
#
TERM=vt100; export TERM
stty intr  c  erase  ?  kill  u  echoe echok

TZ=EST5EDT; export TZ
cat /etc/motd
if mail -e
        then echo "You have mail."
fi
```

Figure 14-2 The Global Profile

2) **The Local Profile**

   The local or individual profile is **.profile** for **sh** users, and **.login** for **csh** users. These files reside in a user's home directory. These local profiles are copies of two prototype files: **/etc/stdprofile** and **/etc/stdlogin**. At the local profile level, users can add commands and variables to customize their environments. A local profile does not have to exist, but if it does exist, it is executed at login time, after the execution of the global profiles **/etc/profile** or **/etc/login.csh**.

Environment Variables

An array of strings called the environment is made available by **exec(2)** when a process begins. Since **login** is a process, the array of environment strings is made available to it. The following local profiles show how environment variables can be defined for users running the shell (**sh(1)**) or the C shell (**csh(1)**).

```
#LOCAL .PROFILE (sh)                    #LOCAL .LOGIN (csh)
#                                       #
#                                       #
LOGNAME=poulet                          set prompt = 'sys5>'
PWD=/usr/poulet                         set noclobber
HOME=/usr/poulet                        setenv PATH .:$HOME/bin:/usr/bin
PATH=:/bin:/usr/bin                     date
SHELL=/bin/sh                           mailx -H
MAIL=/usr/mail/poulet                   echo '  '
```

Figure 14-3 Local Profiles

Other programs make use of the information in the environment array list. New strings can be defined at any time. By convention they are defined with the variable in uppercase letters, followed by an equal sign, followed by the value. The individual **.profile** and **.login** files can contain whatever the user wants.

Default Permissions Mode: umask

A system default controls the permissions mode of any files or directories created by a user. The DG/UX system gives default values of 666 for files and 777 for directories (see **chmod(1M)**). That means that for files everyone automatically gets read and write permission. For directories, everyone gets read, write, and execute permission. (Execute permission on a directory means the ability to **cd** to the directory and to copy files from it.)

Users frequently set up a user mask in their local profiles by means of the **umask(1)** command. **umask** alters the default permission levels by a specified amount. For example:

```
$ umask 027
```

leaves the permission level for owner unchanged, lowers the permission level for group by 2, and reduces the permissions for others to zero. The system default was 666; this user mask changes it to 640, which translates into read and write permission for the owner, read permission for the group, and no permission for others.

A **umask** command in a user's global profile does not change a user's ability to put one in the local profile. The local profile always overrides the global.

Default Shell and Restricted Shell

Generally, when a user logs in the default program that is started is **/bin/sh**. There may be cases, however, where a user needs to be given a restricted shell, **/bin/rsh**. A restricted shell is one where the user is not allowed to:

- change directories
- change the value of $PATH
- specify 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.

User Communication Services

Several ways of communicating with and among users are available. Some of the most frequently used are described in this section.

Message of the Day

You can put items of broad interest that you want to make available to all users in the **/etc/motd** file. The contents of **/etc/motd** are displayed on the user's terminal as part of the login process. The login process executes global files called **/etc/profile** for **sh** users and **/etc/login.csh** for **csh** users. In these files is commonly contained the command:

cat /etc/motd

Any text contained in **/etc/motd** is displayed for each user each time the user logs in. For this information to have any impact on users, you must take pains to use it sparingly and to clean out outdated announcements. A typical use for the Message of the Day facility might be

```
5/30:  The system will be unavailable from 6-11pm Thursday, 5/30
- preventive maintenance.
```

Part of the preventive maintenance should be to remove the notice from **/etc/motd**.

News

Another electronic bulletin board facility is the **/usr/news** directory and the **news**(1) command. The directory stores announcements in text files, the names of which are usually used to provide a clue to the content of the news item. The **news** command displays the items on your terminal.

The **/etc/profile** file can also inform users about news items. A typical **/etc/profile** contains the line:

```
news -n
```

The **−n** argument causes the names of files in the **/usr/news** directory to be printed on a user's terminal as the user logs in. Item names are displayed only for current items, that is, items added to the **/usr/news** directory since the user last looked at the news. The idea of currency is implemented like this: when you read a news item an empty file named **.news_time** is written in your login directory. As with any other file, **.news_time** carries a time stamp indicating the date and time the file was created. When you log in, the system compares the time stamp of your **.news_time** file and time stamp of items in **/usr/news**.

Unlike the Message of the Day where users have no ability to turn the message off, with **news** users have a choice of several possible actions:

read everything	If the user enters the command, **news** with no arguments, all news items posted since the last time the user typed in the command are printed on the user's terminal.
select some items	If the **news** command is entered with the names of one or more items as arguments, only those items selected are printed.
read and delete	After the **news** command has been entered, the user can stop any item from printing by pressing the DELETE key. Pressing the DELETE key twice in a row stops the program.
ignore everything	If the user is too busy to read announcements at the moment, they can safely be ignored. Items remain in **/usr/news** until removed. The item names will continue to be displayed each time the user logs in.
flush all items	If the user simply wants to eliminate the display of item names without looking at the items, a couple of techniques will work:

```
$ touch .news_time
```

updates the time-accessed and time-modified fields of the

```
$ news > /dev/null
```

prints the news items on the null device.

Broadcast to All Users

With the **wall(1M)** command, you can send "broadcast" messages to the screens of all users currently logged on the system. While **wall** is a useful device for getting urgent information out quickly, users tend to find it annoying to have messages print out on their screens right in the middle of whatever else is going on. The effect is not destructive, but is somewhat irritating. It is best to reserve this for those times when you need to ask users to get off the system so that you may do an administrative task.

DG/UX System Mail

The DG/UX system has two electronic mail utilities through which users can communicate among themselves. If your system is connected to others by networking facilities, **mail(1)** and **mailx(1)** can be used to communicate with persons on other systems.

The **mail** program is the basic utility for sending messages. The **mailx** program uses **mail** to send and receive messages, but adds some useful features for storing messages, adding headers, and many other functions. Because of these added features, **mailx** is often the preferred mail facility among users.

Users have the ability, by default, to send and receive mail when you add them to the system with the **adduser** command. A simple example of using **mailx** follows. In a hypothetical world, Poulet wants to send a mail message to Moe. He types:

```
$ mailx moe ↵

  Subject: rubber chicken ↵

Please return my rubber chicken when the puppet show is over.

^D ↵

$
```

Poulet types in a `Subject:` line, hits New Line, and then types the message. When finished, he types Ctrl-D on the next line after finishing the message, and the message is mailed.. The next time Moe presses the New Line key, or when he logs in, his screen will display:

```
You have new mail.
```

Then, Moe types the **mailx** command to see what mail has arrived. For detailed information on using mail facilities, see *Using the DG/UX™ System.*

A setup file called **.mailrc** contains the mail characteristics for each user. You can find a description of how to use a **.mailrc** set-up file in the **mailx**(1) pages of the *User's Reference to the DG/UX™ System.*

User Trouble Log

As the system administrator, you can expect users to look to you to help solve any number of problems. Others in your situation have found it helpful to keep a user trouble log. You will discover that user problems will fall into patterns. And if you keep a record of how these problems are resolved, you will not have to start from scratch every time a problem occurs. The following is an example trouble report that you can use to track and record system problems.

```
************************************************************

                      TROUBLE REPORT

1. Machine ____________________________________________

2. DG/UX Revision _____________________________________

3. Program/command running ____________________________

4. Error Messages _____________________________________

   ____________________________________________________

   ____________________________________________________

5. Symptoms ___________________________________________

   ____________________________________________________



   Date __________

   Person Reporting ______________  Login _____________

   Location ______________________  Phone _____________
************************************************************
```

Figure 14-4 Sample Trouble Report

Expert User Management Information

The remainder of this chapter provide additional information on user services. The **sysadm** program should take care of any questions you have while you are performing the major procedures for user services. Still, you should feel free to read and use this additional information to enhance your understanding and performance.

User Passwords

Before users are permitted to log in to your system, they must be listed in the **/etc/passwd** file. An entry in the **passwd** file consists of a single line with the following seven colon-separated fields:

login_name:password:uid:gid:comment:home_directory:program

For a user named L. Q. Poulet, the line might look like the following:

```
poulet:Rm27oQak1:103:104:L.Q. Poulet:/usr/poulet:/bin/csh
```

The fields are

login name	`poulet` : A valid name for logging onto the system. A login name can be up to eight alphabetic or numeric characters; the first character must be alphabetic. It is usually chosen by the user.
password	`Rm27oQak1`: A user chooses a password and registers it with the system with the **passwd(1)** command. The encrypted form of the password appears in this field. No one but the user ever knows the real password. The actual password can be a maximum of eight characters. At least one character *must* be a numeric character or special character. This is to discourage users from choosing ordinary words as passwords. When you add a user to the file you may use a default password, such as **passwd9**, and instruct the user to change it at the first login. Following the encrypted password, separated by a comma, there may be a field that controls password aging. For information on password aging see **passwd(4)** and **passwd(1)**.
user ID	`103`: The user ID number (**uid**) is between 101 and 60,000. The number must not include a comma. Numbers 100 and below are reserved. User ID 0 is reserved for the superuser.

group ID	`104`: The same conditions apply to the group ID (**gid**) number as to the **uid**.
comment	`L.Q.Poulet`: Optional. May contain user's name, office, office phone number, home phone number, etc. There is no required format for this field.
home directory	`/usr/poulet`: The directory where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory such as **/usr**. The home directory is the origination point of the user's directory tree.
program	The name of a program invoked at the time the user logs in. If the field is empty, the default program is **/bin/sh**. This field is most commonly used to invoke a special shell, such as **/bin/rsh** (restricted shell).

As noted above, the password field may contain a subfield that controls the aging of passwords. A description of how the process works can be found in **passwd**(4). The effect is to force users periodically to select a new password. If password aging is not used, a person can keep the same password indefinitely.

Group IDs

Group IDs are a means of establishing another level of ownership of and access to files and directories. Users with some community of interest can be identified as members of the same group. Any file created by a member of the group carries the group ID as a secondary identification. By manipulating the permissions field of the file, the owner (or someone with the effective user ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the **/etc/group** file. Entries consist of the following colon-separated fields:

group_name:password:gid:login_names

A sample entry from this file is shown and explained below:

```
prog::104:reynard, poulet
```

Each entry is one line; each line has the following fields:

group name	`prog`: The group name can be up to eight characters, the first of which must be alphabetic.
password	The password field should not be used.

group ID	`104`: The group ID is a number from 101 to 60,000. The number must not include a comma. Numbers below 100 are reserved.
login names	`reynard, poulet`: The login names of group members are in a comma-separated list. A login name should be a member of no more than one group. There is nothing to prevent a user from having more than one login name, however, as long as each is unique within the system.

Password Aging

The password aging mechanism forces users to change their passwords periodically. or prevents them from changing password before a specified time interval. Password aging is selectively applied to logins by editing the **/etc/passwd** file.

The password aging information is appended to the encrypted password field in the **/etc/passwd** file. The password aging information consists of a comma and up to four characters in the format:

,Mmww

The meaning of these fields is as follows:

,	The delimiter between the password itself and the aging information.
M	The **M**aximum duration of the password in weeks.
m	The **m**inimum time interval before the existing password can be changed by the user, in weeks.
ww	The week (counted from the beginning of 1970) when the password was last changed. You add this information through the codes that you edit into the **/etc/passwd** file. Then, the system automatically adds these characters to the password aging field. All times are specified in weeks (0 through 63) by a 64 character alphabet.

Figure 14-5 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.

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

Figure 14-5 Password Aging Character Codes

Two special cases apply for the character codes:

- If *M* and *m* are equal to zero (the code, **..**), the user is forced to change the password at the next login. No further password aging is then applied to that login.
- If *m* is greater than *M* (for example, the code, **./**), only **root** is able to change the password for that login.

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 (/).

1) Here is a typical login/password entry in the **/etc/passwd** file for the typical user **jqu**:

```
jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:
```

2) To cause **jqu** to change the password at least every 2 weeks, but keep it at least for 1 week, you should use the code **0/**. After you edit the **/etc/passwd** file, adding **,0/** to the password field, the entry looks like this:

```
jqu:RTKESmMOE2m.E,0/:100:1:J. Q. Username:/usr/jqu:
```

After the password entry is changed, **jqu** will have to change the password at the next login and every 2 weeks thereafter.

3) After **jqu**'s first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field.

```
jqu:RTKESmMOE2m.E,0/W9:100:1:J. Q. Username:/usr/jqu:
```

In this example, **jqu** changed the password in week **W9**.

Changing or Deleting Aliases

As with changes to the **/etc/passwd** file, all changes that you make when adding or deleting an alias by hand in the **/usr/lib/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 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 **/usr/lib/aliases**, run the following command:

```
# /usr/lib/sendmail -bi ↵
```

This command initializes the alias database, making your latest changes available to the system.

End of Chapter

Chapter 15
Accounting Management

The DG/UX accounting system is a collection of C language programs and shell procedures with which you can monitor how system resources are being used. System-use data is logged and then organized and directed into summary files and reports which you can print or display on your terminal. Among other things, these reports are useful for helping to maintain security because you can trace who logged on when and what commands were run.

The accounting functions are not handled by the **sysadm** program.

The major sections of this chapter are:

- What the Accounting System Does
- The Default Accounting System
- How the Accounting System Works
- Daily Accounting with Runacct
- Runacct Reports
- Recovering From Failure
- Updating Holidays
- Directories and Files

What the Accounting System Does

The accounting system does the following:

- Records connect sessions, logins, date changes, reboots, and shutdowns.
- Records disk use and system use for each login.
- Daily: formats accounting data into summary files and reports.
- Monthly: saves summary files, generates a report, and cleans up files.

The Default Accounting System

To make accounting easier to use, the DG/UX system comes with a ready-to-go method of doing your accounting. This section describes this accounting system. The remainder of this chapter shows you in detail how the components of the accounting system function.

When you bring the system to run level 3, the accounting system will do the following:

- Run the **runacct** program daily at 4 a.m.
- Direct error messages to **fd2log**.
- Run **dodisk** daily at 2 a.m.
- Run **ckpacct** at 5 minutes after every hour.
- Run **monacct** at 5:15 a.m. on the first day of every month.
- Clean out files at 3 a.m. in **/tmp** that are older than three days.

The above actions are executed by the following lines in **/usr/spool/cron/crontabs/root.proto**:

```
0 4 * * * /bin/su - adm -c "/usr/lib/acct/runacct
      2>  /usr/adm/acct/nite/fd2log"
0 2 * * * /usr/lib/acct/dodisk
5 * * * * /bin/su - adm -c "/usr/lib/acct/ckpacct"
15 5 1 * * /bin/su - adm -c /usr/lib/acct/monacct
0  3 * * 2-5 /bin/find /tmp -atime +3 -exec rm {} \;
```

The **monacct** procedure cleans up all daily reports and daily total accounting files and deposits one monthly total report and one monthly total accounting file in the **fiscal** directory. The default action of **monacct** adds the current month's date to the filenames.

To make these lines available to **cron**, you must type:

```
# cd /var/spool/cron/crontabs ↵
# cat root.proto >> root ↵
# crontab root ↵
```

Changing the Default Accounting System

If you wish to change the default accounting system, edit **/var/spool/cron/crontabs/root**. Any time you make changes to this file, you must run the **crontab(1)** command on it.

To Turn Accounting On

Finally, before your accounting system will work, you must do two things:

1) Edit **/etc/dgux.params**. Change **account_START="false"** to **account_START="true"**.

2) Type: **# /usr/lib/acct/turnacct on** ⮐

   If you should decide not to use the accounting system, you may turn it off by specifying "off" as an argument.

To Print Default System Daily Reports

The default accounting system produces these reports. If you type the following command line, you will display the reports assembled for April 11. Without a date designation, **prdaily** prints out the previous days's reports. You can also direct the output to a file or lineprinter.

```
# /usr/lib/acct/prdaily 0411 ⮐
```

To Print Default System Monthly Reports

The default accounting system produces these reports. Monthly reports are in **/usr/adm/fiscal/fiscrpt***xx*, where *xx* is the number of last month, such as 01 for January. You can direct the contents of these files to your screen or to a lineprinter.

How the Accounting System Works

The accounting system may be thought of as having three major parts: a logging mechanism, a scheduling mechanism, and a processing mechanism for the data that is logged.

- The general *logging mechanism* is activated by a script called **/usr/lib/acct/startup**; it does the major logging work for all user processes running. The **startup** script is executed according to a setting in **/etc/inittab**. Normally, the **startup** script is set to begin running in multi-user run levels 2 or 3. In this case, whenever your system enters run levels 2 or 3, the **startup** script is executed, thus the logging mechanism for the accounting system automatically begins working in run levels 2 or 3 (or whatever you set). Chapter 4 explains how the **init(4)** program reads entries from **/etc/inittab** to establish run levels.

- The *scheduling mechanism* is the **cron(1M)** daemon. You use **cron** as a timer to start the accounting programs at regular intervals. We demonstrated how this scheduling mechanism works earlier in "The Default Acccounting

System."

- The *processing mechanism* is a set of accounting programs in **/usr/lib/acct** that you specify for **cron** to execute.

Accounting Programs

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 Programs

You can invoke the following programs from the shell.

turnacct	An interface to the **accton** program that turns process accounting on or off. When switched "on", **turnacct** starts the **accton** program; "off" stops the **accton** program. When "off", the accounting system is disabled.
acctcom	Searches and prints process accounting files. Note that this command is in **/usr/bin**.
chargefee	Records billing information for file restores and other services and writes the data to **/usr/adm/fee**. This data is picked up by the **runacct** program and merged into the total accounting records.
prdaily	Displays the previous day's report.
wtmpfix	Checks **/var/adm/acct/nite/wtmp** as step 2 in the execution of the **runacct** program. See "Daily Accounting with Runacct" later in this chapter for details on **runacct**.
fwtmp	Fixes corrupted accounting files by manipulating content from binary to ASCII and back. See "Fixing Corrupted Files" later in this chapter.
acctcms	Produces a summary of all processes that execute commands. See **acctcms(1M)**.
acctprc1	Reads input in the form described by **acct(4)** and adds login names corresponding to user IDs. Then, it writes an ASCII line for each process giving user ID, login name, prime CPU time (ticks), non-prime CPU time (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 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 **runnact** and writes it into a monthly summary. Invoke **monacct** via **cron** once each month or each accounting period. **Monacct** creates summary files in **/var/adm/acct/fiscal**. Table 5-4 shows how **monacct** organizes this data. See **acctsh(1M)**.
dodisk	Does disk accounting on the special files in **/etc/fstab**. Creates **/var/adm/dtmp**.
diskusg	Generates disk accounting information. See **diskusg(1M)**.
acctcon1	Reads **/etc/wtmp** and converts login/logoff data to a sequence of records, one per login session. The output is ASCII, giving

	device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time, and starting date and time. See **acctcon(1M)** for complete details.
acctcon2	This program expects a sequence of login session records as input and converts them into total accounting records (**tacct**). See **acct(4)** for the format of **tacct**.
acctdisk	Reads lines from **/var/adm/dtmp** (which is created by **dodisk**) containing user ID, login name, and number of disk blocks, and converts them to total accounting records that are merged with other accounting records. See **acct(1M)**.
acctdusg	Computes disk resource consumption (including indirect blocks) for each login. See **acct(1M)**.
acctwtmp	Called by the **shutdown(1M)** command. Writes a **utmp(4)** record containing the current time and a string of characters describing "reason" which is either accton or acctoff.
acctmerg	Merges total accounting files (**tacct**). See **acctmerg(1M)**.
lastlogin	Invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date each person logged in.
nulladm	Called by most accounting scripts. Creates files with mode 664 and ensures that owner and group are **adm**.
shutacct	Invoked automatically during system shut down (via **/etc/shutdown**) to turn process accounting off and append a "reason" record to **/etc/wtmp**.

The data organized by the accounting programs is manipulated, stored, and cleaned up through interactions with files in the following directories:

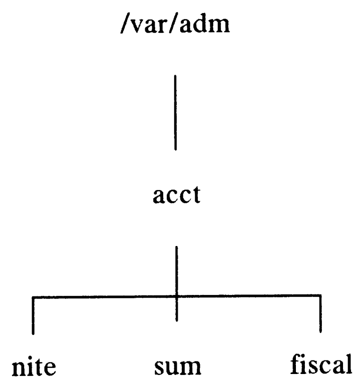


Figure 15-1 The Accounting Directories

The **/var/adm** directory contains the data collection files. The **nite** directory contains files that are reused daily by the **runacct** procedure. The **sum** directory contains the cumulative summary files updated by **runacct**. The **fiscal** directory contains periodic summary files created by **monacct**.

Daily Accounting with Runacct

Runacct(1M) is the main daily accounting shell procedure. It is normally started by the **cron(1M)** program. **Runacct** works on files that contain data on connects, fees, disk usage, and accounting processes. It also prepares daily and cumulative summary files for use by the **prdaily** program or for billing purposes. **Runacct** programs produce the following files:

nite/lineuse	Produced by **acctcon(1M)**, which reads the **wtmp** file, and produces usage statistics for each terminal line on the system. This report is useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3:1, the line is probably failing.
nite/daytacct	Yesterday's total accounting file in binary format.
sum/tacct	The accumulation of each day's **nite/daytacct** procedure ; it can be used for billing. The **monacct** shell procedure restarts the file each month or fiscal period.
sum/daycms	Produced by the **acctcms** program; contains the daily command summary. The ASCII version of this file is **nite/daycms**.
sum/cms	The accumulation of each day's command summaries. Restarted by the execution of **monacct**. The ASCII version is **nite/cms**.
sum/loginlog	Produced by the **lastlogin** shell procedure. Maintains a record of the last time each login was used.
sum/rprt.MMDD	Each execution of **runacct(1M)** saves a copy of the output of the **prdaily** shell procedure in this file.

Runacct(1M) does not damage files if errors occur. It tries to recognize errors, provide diagnostics, and terminate processing so that it can be restarted easily. **Runacct** writes records of its progress into a file named **active** and uses files in the **/var/adm/acct/nite** directory unless otherwise noted. By default, all diagnostics from a **runacct** are directed into the **/var/adm/acct/nite/fd2log** file.

When **runacct** starts up, its run is protected by lock files which cause any multiple invocations of the program to terminate. These lock files in the **nite** directory are named **lock** and **lock1**. The **lastdate** file contains the month and day **runacct** was last invoked and prevents more than one execution per day. If **runacct** detects an error, it writes a message to **/dev/console**, sends mail to **root**, removes the locks, saves the diagnostic files, and terminates execution.

So that **runacct** can be restartable, processing is broken down into separate reentrant states, using a case statement inside a *while* loop. Each state is one instance of the **case** statement.

When each state completes, **/nite/statefile** is updated to reflect the next state. In the next loop through the **while**, **statefile** is read and the case falls through to the next state. At the CLEANUP state, **runacct** removes the locks and terminates. States are executed in the following order:

SETUP	Move active accounting files into working files. The command **turnacct switch** is executed. The process accounting files, **/var/adm/pacct?**, are moved to **/var/adm/Spacct?.MMDD**. The **/etc/wtmp** file is moved to **/var/adm/acct/nite/owtmp**.
wtmpfix	Verify the integrity of **/etc/wtmp**, correcting date changes if necessary. The **wtmp** file in the **nite** directory is checked by the **wtmpfix** program. Some date changes cause the **acctcon1** shell procedure to fail, so **wtmpfix** tries to adjust the time stamps in the **wtmp** file if a date change record appears.
CONNECT1	Produce connect session records. Connect session records are written to **ctmp** in binary form. The **lineuse** file is created, and the **reboots** file is created showing all of the boot records the **wtmp** file.
CONNECT2	Convert session records into total accounting records. **Ctmp** is converted to **ctacct.MMDD** which contains accounting records.
PROCESS	The **acctprc1** and **acctprc2** programs convert the process accounting files, **/var/adm/Spacct?.MMDD**, into total accounting records in **ptacct?.MMDD**. The **Spacct** and **ptacct** files are correlated by number so that if **runacct(1M)** fails, **Spacct** files are not reprocessed. *Note*: When restarting **runacct** in this state, remove the last **ptacct** file; it 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/tacctMMDD**, 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/rprtMMDD**, remove the locks, then exit.

At the completion of the last state, an ASCII file exists in directory **sum**; it is labeled with today's date, such as **rprt1120** 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 are in directory **fiscal**. The report for November is be **fiscrpt11**.

Runacct Accounting Reports

Runacct generates the following reports:

- Daily line usage
- Daily usage by login name
- Daily command summary
- Monthly command summary
- Last login

You cannot generate an individual report, say one just for today's line usage. All five reports are generated at once. Sample reports and meanings of their data follow.

Daily Line Usage

The first part of the report is the **from/to** banner. The banner displays the time the last accounting report was generated and the time the current accounting report is generated. It is followed by a log of system reboots, shutdowns, recoveries, and any other record dumped into **/etc/wtmp** by the **acctwtmp** program.

The second part of the report is a breakdown of line usage. TOTAL DURATION tells how long the system was accessible through the terminal lines. The columns are:

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 on.
# Off	Number of times a user logged off, and any interrupts on that line. Generally, interrupts occur on a port when **getty(2)** is first invoked when the system goes to multi-user mode. Interrupts occur at a rate of about two per event. Therefore, you often see more than twice the amount of OFF than ON or SESS. If the # OFF exceeds the # 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** since 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.

The following is an example daily line usage report:

Table 15-1 Line Usage Report

```
Nov 11 16:00 1989  DAILY REPORT For DG/UX page 1

from Tue Nov 10 08:27:00 1989 to   Wed Nov 11 04:00:42 1989

2    shutdown

1    runacct

Total Duration is 1174 Minutes
```

Line	Minutes	Percent	# Sess	# On	# Off
tty01	0	0	1	1	0
tty02	506	43	1	1	1
tty03	48	4	5	5	5
console	41	3	1	1	1
TOTALS	693	--	9	9	8

Daily Usage by Login Name

This report gives a by-user breakdown of system resource use. Its data consists of:

UID	User ID.
LOGIN NAME	Login name of the user. (There can be more than one login name for a single user ID.)
CPU (MINS)	CPU time used, divided into PRIME and NPRIME (nonprime). See "Updating Holidays," later in this chapter.
KCORE-MINS	Total memory a process used, in kilobytes per minute. Divided into PRIME and NPRIME.
CONNECT (MINS)	How long a user was logged into the system, by PRIME and NPRIME time. If this time is long and the column #OF PROCS is low, this user rarely uses the terminal.
DISK BLOCKS	When the disk accounting programs have been run, their output is merged into the total accounting record (**tacct.h**) and shows up in this column. **Acctdusg** does disk accounting.
# OF PROCS	Number of processes invoked by the user. Large numbers may indicate that a shell procedure looped.
# OF SESS	Number of times the user logged onto the system.
# DISK SAMPLES	Number of times the disk accounting was run to obtain the average number of disk blocks.
FEE	Total accumulation of total charges against the user. The **chargefee** procedure charges for special services, such as file restores and tape manipulation by operators. This process is low-level and will track whatever units you define.

The following is a sample by-user report.

Table 15-2 Daily Usage by Login Name

Nov 11 04:42 1989 Daily Usage Report

Uid	Login	CPU (Min)		Kcore-Mins		Connect(Min)		Disk	# Of	# Of	# Disk	
	Name	Prime	NPrime	Prime	NPrime	Prime	NPrime	Blocks	Procs	Sess	Samples	Fee
0	TOTAL	37	235	1198	4671903	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	3974150	0	0	1	0
3	sys	0	0	0	0	0	0	0	0	1	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

Daily and Monthly Total Command Summaries

These two 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, which is a good way to calculate drain on a system.

COMMAND NAME	The name of the command. All shell procedures are reported under the name **sh**. Monitor the frequency of programs called **a.out**, or **core**, or any other program that does not conform to the DG/UX command structure. **Acctcom** is also a good way to determine who executed an incorrect command and if superuser privileges were used.
NUMBER CMDS	Number of times a command was invoked.
TOTAL KCOREMIN	Total Kbyte segments of memory used by a process, per minute of run time.
TOTAL CPU-MIN	A program's total processing time.
	Total real-time minutes this program has run.
MEAN SIZE-K	Mean of the TOTAL KCOREMIN divided by TOTAL CPU-MIN.
MEAN CPU-MIN	Mean of the NUMBER CMDS and TOTAL CPU-MIN.
HOG FACTOR	Ratio of system availability to system usage: (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. This column may be negative.
BLOCKS READ	Total physical block reads and writes that a process performed.

Examples of daily and monthly command summaries follow.

Table 15-3 Daily Command Summary

Nov 11 04:42 1989 Daily Command Summary

Command Name	Numb Cmds	Total Koremin	Total CPU-Min	Total Real-Min	Mean Size-K	Mean CPU-Min	Hog Factor	Chars Trnsfd	Blocks Read
TOTALS	2332	1624.74	16.05	15210.91	5.67	003	0.01	0	0
nroff	1	249.86	2.03	2.91	123.19	2.03	0.70	0	0
troff	3	305.12	2.60	3.10	411.11	2.89	0.99	0	0
sh	1028	434.53	7.24	7632.44	59.99	0.01	0.01	0	0
csh	1115	474.13	3.96	6534.53	41.111	0.01	0.40	0	0
sendmail	18	55.79	0.93	2.10	155.96	0.04	0.26	0	0
ls	111	62.69	0.57	4.35	98.99	0.01	0.21	0	0
more	35	25.43	0.19	207.16	84.93	0.06	0.00	0	0
ps	1	20.74	0.63	0.35	173.62	0.14	0.33	0	0
cp	20	317.311	2.94	3.41	339.97	0.09	0.21	0	0

Table 15-4 Monthly Command Summary

Nov 11 04:42 1989 Monthly Total Summary

Command Name	Numb Cmds	Total Koremin	Total CPU-Min	Total Real-Min	Mean Size-K	Mean CPU-Min	Hog Factor	Chars Trnsfd	Blocks Read
TOTALS	27792	17118.47	227.94	77021.94	1122.74	3.71	0	0	0
nroff	30	4216.86	34.03	58.99	123.19	1.03	0.70	0	0
troff	23	3105.12	25.60	43.10	411.11	1.89	0.99	0	0
sh	13118	5551.53	93.16	50386.06	59.59	0.01	0.01	0	0
csh	10115	3474.13	33.96	26534.53	41.11	0.01	0.40	0	0
sendmail	238	1455.79	8.93	52.10	165.97	0.03	0.16	0	0
ls	1075	1062.21	9.57	54.85	113.99	0.01	0.17	0	0
more	185	825.43	6.19	107.16	124.93	0.06	0.00	0	0
ps	38	520.74	3.63	11.35	181.72	0.08	0.50	0	0
cp	2970	384.31	8.94	52.85	43.93	0.00	0.17	0	0

Last Login

This report gives the date when a particular login was last used. The report can help you find likely candidates for the tape archives, such as **/usr** directories associated with unused login names.

Table 15-5 Last Login Report

```
Nov 11 04:42 1989 LAST LOGIN

          ________________________________________
          00-00-00 bin       89-11-02 carson

          00-00-00 croot2    89-10-09 moe

          00-00-00 daemon    89-11-11 larry

          00-00-00 svvs      89-11-10 curly

          00-00-00 sync      89-11-05 leah

          00-00-00 archive   89-11-01 tlp

          87-11-12 poulet    89-11-11 uucp
          ________________________________________
```

A field of all zeros means that login has not been used since the last invocation of the **lastlogin** program.

Recovering from Failure

If the system crashes, **/usr** runs out of space, or a **wtmp** file is corrupted, **runacct** fails. If the **activeMMDD** file exists, check it first for error messages. If the **active** file and **lock** file exist, check **fd2log** for error messages.

Runacct produces the following error messages. We suggest ways to recover from them.

ERROR: `locks found, run aborted`

The files **lock** and **lock1** were found in **/var/adm/acct/nite**. Remove these files before restarting **runacct**.

ERROR: `acctg already run for date: check /var/adm/acct/nite/lastdate`

The date in **lastdate** and today's date are the same. Remove lastdate.

ERROR: `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)**.

ERROR: `Spacct?.MMDD already exists`

File setups have already run. Check status of files, then run setups manually. See the following section, "Restarting Runacct."

ERROR: `/var/adm/acct/nite/wtmp.MMDD already exists, run setup manually.`

See "Fixing Corrupted Files" later in this chapter.

ERROR: `wtmpfix errors see /var/adm/acct/nite/wtmperror`

Wtmpfix detected a corrupted wtmp file. Use **fwtmp** to fix the file.

ERROR: `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.

Restarting Runacct

Runacct called without arguments assumes this is the first invocation of the day. You must use the argument MMDD if **runacct** is being restarted; MMDD specifies the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of **statefile**. To override **statefile**, include the desired state on the command line. As we said earlier, **runacct** is normally started by **cron**. But should you need to start **runacct** from the command line, here are three ways you might do it:

To start **runacct**, type:

```
# nohup  runacct 2>  /var/adm/acct/nite/fd2log &
```

To restart **runacct** specifying MMDD (0601), type:

```
# nohup  runacct  0601 2>  /var/adm/acct/nite/fd2log &
```

To restart **runacct** at a specific state, such as wtmpfix, type:

```
# nohup  runacct  0601  wtmpfix  2> /var/adm/acct/nite/fd2log &
```

In the above examples, the **2** sends the standard error output to the file named **f2dlog**; check this file periodically for error messages. Specifying MMDD creates a new **active0601** file, dated June 1.

Fixing Corrupted Files

Sometimes, a file is corrupted or lost. Some files can be restored from backup tapes. However, you must fix certain files.

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, **wtmpMMDD** is created. The following procedure shows you how to fix this file. At the editing step below, you'll see binary lines mixed in with ASCII lines. Fixing this file consists of deleting the binary lines. Here's the procedure:

1) Go to directory **/var/adm/acct/nite**.

2) Type: **fwtmp < wtmpMMDD > xwtmp** ↵

   This executes the **fwtmp(1M)** program on the contents of **wtmpMMDD**, and stores the output in file **xwtmp**. So what you are doing here is converting the binary contents of **wtmpMMDD** into ASCII format.

3) Now, edit **xwtmp** and delete all binary lines.

4) When you've finished editing, convert from ASCII back to binary:

   Type: **fwtmp -ic < xwtmp > wtmp.MMDD** ↵

If the **wtmp** file is beyond repair, create a null **wtmp** file. This will prevent any charging of connect time.

Fixing tacct Errors

If you are using the accounting system to charge users for system resources, the integrity of **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 **sum/tacctprev** with **prtacct**. If **prtacct** does not report any errors, patch up **sum/tacct.MMDD** and recreate **sum/tacct**. A simple patch-up procedure is:

1) Go to directory **/var/adm/acct/sum**.

2) Type: **acctmerg -v < tacct.MMDD > xtacct** ↵

   This executes the **acctmerg(1M)** program on the contents of **tacct.MMDD**, and stores the output in file **xtacct**. So what you are doing is converting the binary contents of **tacct.MMDD** into ASCII format.

3) Now, edit **xtacct** and delete all corrupted records.

4) When you've finished editing, convert from ASCII back to binary. Type:

   acctmerg -i < xtacct > tacct.MMDD ↵

Remember that **monacct** removes all the **tacct.MMDD** files; therefore, when you merge these files together you recreate **sum/tacct**.

Updating Holidays

The file **/usr/lib/acct/holidays** contains the prime/non-prime table for the accounting system. Edit the table to reflect your holiday schedule for the year. The table format has three types of entries:

1) *Comment Lines:* Comment lines can appear anywhere in the file as long as the first character in the line is an asterisk.

2) *Year Designation Line:* This line should be the first data line (noncomment line) in the file; it must appear only once. It has three fields: year, prime time, and non-prime time. For example, to specify the year 1989, prime time at 9:00 a.m., and non-prime time at 4:30 p.m., enter:

   ```
   1989  0900  1630
   ```

   The time 2400 is automatically converted to 0000.

3) *Holiday Lines:* These entries follow the year designation line, and have the format:

   day-of-year Month Day Description of Holiday

   The day-of-year field is a number between 1 and 366, indicating the day for the corresponding holiday; leading blanks, tabs, and spaces are ignored. The other three fields are commentary, and are not currently used by other programs.

The accounting system will not function properly unless a **holidays** file exists. You will receive a **/usr/lib/acct/holidays.proto** as part of the DG/UX system. When you boot the system for the first time, **holidays.proto** will automatically be copied to **holidays**. Add entries to **holidays** as suits your needs. A sample file follows:

```
* Prime/Nonprime Table for DG/UX Accounting System
*
* Current     Prime     Nonprime
*  Year       Start     Start
*
  1989        0900      1630
*
* Day of      Calendar  Holiday
*  Year         Date
*
   1          Jan 1     New Year's
   2          Jan 2     day after
 152          May 31    Memorial Day
 186          Jul 4     Independence Day
 329          Nov 25    Thanksgiving
 358          Dec 24    Christmas
```

Accounting Directories and Files

This last sections of the chapter briefly describes all the files within the DG/UX accounting structure.

Files 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

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**.
logMMDD	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.
wtmperrorMMDD	Same as **wtmperror** after **runacct** detects an error.
wtmp.MMDD	Previous day's **wtmp** file.

Files 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

cms	Total command summary file for a fiscal period.
fiscrpt	Report similar to **prdaily** for a fiscal period.
tacct	Total accounting file for a fiscal period.

End of Chapter

Appendix A
Device Names and Codes

This appendix explains the different forms that nodes take for all devices that the DG/UX system can use, including terminal controllers and lines, disks, tape drives, line printers, and pseudo-devices. The **/usr/etc/master.d** directory contains files that list *all* devices, pseudo-devices, protocols, and configuration parameters that are supported by the DG/UX system and any other packages you have installed. The **/usr/src/uts/aviion/Build/system.aviion** file describes *your* system's devices.

Accessing Devices

A node describes a logical device. It creates a link between a physical medium and a logical unit accessed by a program. When you boot your system, the kernel creates all the necessary nodes in **/dev**, based on the entries in your system file. Every time you reboot your system, the kernel deletes all nodes in **/dev** and creates new nodes that correspond to all the devices configured in your kernel (described in the system file) and present on your system. Because nodes are removed at boot time, we recommend that you do not create them manually.

When a node is created at boot time, the kernel associates a node name with information including a major number, a minor number and whether the file is to be accessed as a block or character device. A **major number** tells the kernel what type of device is represented by the device file. Major numbers are assigned by Data General; they are listed in the master file in **/usr/etc/master.d/dgux**. You can change them by editing the master file, but we don't recommend that you do so unless you're writing your own device driver.

Minor numbers are allocated by the kernel. As administrator, you never have to worry about specifying a minor number. We explain them here so that you will know how the system uses them. A minor number tells the kernel exactly which physical or logical device unit is specified. For example, with terminal controllers, the minor numbers represent the tty lines. For logical disks, minor numbers represent the order in which logical disks have been added to the system, beginning with 0. Tapes are an exception. The minor number for a tape specifies which unit *and* what flags to apply to that tape (rewind, no rewind, density).

The major number is an index to a table in the kernel. This table contains sets of I/O subroutines called **device drivers**. Device drivers are either character-oriented or character-and-block-oriented so that a program can access a device in either character or block mode. For each type of device, there is a driver that invokes open, close, read, and write commands. Programs which need to do I/O call the kernel; the kernel invokes device drivers which use minor numbers to access a particular unit or device.

For disks, **/dev** has subdirectories corresponding to character (raw) and block driver access for each device. Tape devices have only character-special nodes which are in **/dev/rmt**. Physical disks are defined in **pdsk** for block special and in **rpdsk** for character special. Logical disks are defined in **rdsk** (raw disk); and **dsk** (block disk).

The following list recaps node names, major numbers, minor numbers, and device codes:

- *node name* -- This is the name through which the device will be accessed. Nodes for some devices have names that include the hardware device code and physical unit number. You may link other names to the node with the **ln(1)** command.

- *major number* -- The major number identifies the type of device. Every node that refers to a device of a given type will be created with the same major number. For example, all disk units on all **cied** type disk controllers have the same major number. Separate nodes are made for block access and character access to the same device; the major number is the same for each type of access.

- *minor number* -- The minor number points to a specific device or location to be addressed. Within a given type of device, minor numbers start at 0 and proceed sequentially across all devices in that type.

  Tape drives are an exception: their minor number is derived from information that includes the unit number, density, and rewind option required.

- *device code* -- To access a device, the operating system must be able to uniquely identify the device. This is done with device codes. Device codes are assigned by the operating system to all devices that post interrupts. Default device codes are listed later in this appendix.

As Figure A-1 illustrates, devices are connected to the system through device controllers and by direct line.

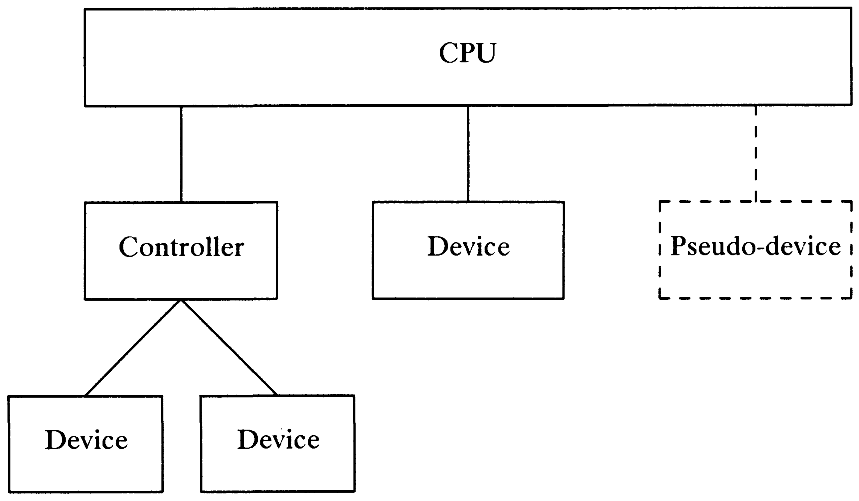


Figure A-1 Device Connections

Pseudo-devices are software constructs that the system uses as though they were devices. Some examples of pseudo-devices are:

/dev/mem	Physical Memory interface.
/dev/kmem	Kernel memory interface.
/dev/null	The null device.
/dev/error	Error recording.
/dev/prf	Operating system profiler.
/dev/tty	Generic terminal interface; associates a tty line with a **login** process.

System File Device Mnemonics

The following devices are supported by the DG/UX system:

con	The system console. This can be either a CRT or a hardcopy console.
syac, sdcp	Systech terminal controller boards: **syac** is asynchronous, **sdcp** is synchronous.
st	SCSI tape
sd	SCSI disk
cisc	Ciprico SCSI adapter
cied	Ciprico ESDI controller: specified via the **cird** or **cied** driver in the system file.
cimd	Ciprico SMD controller: specified via the **cird** or **cimd** driver in the system file.
insc	Integrated SCSI adapter
hken, inen	Intelligent LAN (local area network) controllers. These provide a communications interface, which is used by DG TCP/IP (DG/UX). **hken** is an Interphase Hawk controller. **inen** is an integrated controller.
duart	dual universal asychronous receiver transmitter; a serial communications device
meter	A pseudo-device that is used to monitor various processes in the kernel; not used by system administrators or general users.
pts, ptc	Pseudo terminals used by **telnet(1)**, **shl(1)**, and other applications. For each pseudo terminal connection, a master (**ptc**) and a slave (**pts**) are required.
prf	The operating system profiler, which provides access to system activity information. See **prf(7)**.

Specifying Devices

A device specification names the physical devices by which your machine communicates with peripherals, such as remote disks and tapes. A device specification contains a device driver name followed by a parenthesized list of optional parameters. The device name identifies the type of controller or device; the parameters provide additional information required to fully specify the peripheral.

The first optional parameter specifies a particular controller in configurations which may contain more than one controller of a given type. The second parameter almost always specifies a unit number for those controllers which support multiple units. Null or missing parameters, such as the first parameter in the specification **cied(,1)** or the second and subsequent parameters in **cied(0)** are interpreted as default values. The defaults are interpreted by the device driver itself; by convention, all numerical parameters default to zero. An asterisk as a parameter represents all possible values for the parameter.

The devices you use will depend on your hardware. For instance, AViiON workstations will use the integrated devices (inen, insc), but AViiON servers will not.

Device driver mnemonics are generally constructed from the first two letters of the manufacturer (Ciprico) or the manufacturer's name for the device (Hawk). The second two letters of the mnemonic usually stand for a device type specifier, such as *ed* for ESDI disk. Thus, **cied** represents the device driver program that allows a Ciprico ESDI disk to communicate with the operating system.

The parameter values are all zero-based. Therefore, **cied(1,2)** specifies the third drive of the second Ciprico ESDI controller in the system. The first drive on a system's first ESDI controller would be **cied()**.

AViiON System Specifications

The following tables show the specifications for devices on AViiON Systems. Tables A-1 and A-2 show the default entries for devices on the primary VME bus and the secondary SCSI bus. Note that the lines with entries such as cied@code stand for the nondefault situation. That is, when you have used the first and second defaults for cied, you will have to jumper your own device codes and select base addresses.

Tables A-3 and A-4 show the default base addresses for the entries in Tables A-1 and A-2. The lines with "set" entries mean that there are no default device codes or base addresses for these occurrences of the given device. You must set them yourself. Refer to your hardware documentation for details on setting your own device codes and base addresses.

Table A-1 Primary Bus (VME) AViiON Systems Device Specifications

#	Device Name	Controller Number	Unit	3rd Param	Device Specification
1st	cied	0	0-3	-	cied(0,0)
2nd	cied	1	0-3	-	cied(1,0)
3rd	cied@code	address	0-3	-	cied@code(address,unit)
1st	cimd	0	0-3	-	cimd(0,0)
2nd	cimd	1	0-3	-	cimd(1,0)
3rd	cimd@code	address	0-3	-	cimd@code(address,unit)
1st	sdcp	0	0-1	-	sdcp(0,0)
1st	syac	0	0-15	-	syac(0,0)
2nd	syac	1	0-15	-	syac(1,0)
3rd	syac	2	0-15	-	syac(2,0)
4th	syac	3	0-15	-	syac(3,0)
5th	syac@code	address	0-15	-	syac@code(address,unit)
1st	hken	0	*ether_addr*	-	hken(0,*ether_addr*)
2nd	hken	1	*ether_addr*	-	hken(0,*ether_addr*)

Table A-2 Secondary Bus (SCSI) AViiON System Device Specifications

SCSI Devices

Device Name	Controller Number	ID	3rd Parameter	Device Specification
sd	See cisc below	0-3	-	sd(cisc (),0)
st	See cisc below	4-6	file	st(cisc (),0)

SCSI Adapters

Device Name	Adapter	ID	3rd Parameter	Device Specification
1st cisc	0	-	-	cisc()
2nd cisc	1	-	-	cisc(1)

Table A-3 Default Base Addresses for AViiON System VME Devices

#	Device Name	Device Code	Base Address
1st	cied	18	FFFFEF00
2nd	cied	19	FFFFF100
3rd	cied@code	set	set
1st	cimd	18	FFFFEF00
2nd	cimd	19	FFFFF100
3rd	cimd@code	set	set
1st	sdcp	50	55B00000 and 55B01000
1st	syac	60	60000000
2nd	syac	61	60020000
3rd	syac	62	60040000
4th	syac	63	60060000
5th	syac@code	set	set
1st	hken	15	FFFF4000 and 55900000
2nd	hken	16	FFFF5000 and 55980000

Table A-4 Default Base Addreses for AViiON System SCSI Devices

#	Device Name	Device Code	Base Address
1st	cisc	28	FFFFF300
2nd	cisc	29	FFFFF500

AViiON Station Specifications

The tables in this section show the devices relevant to the DG/UX operating system running on AViiON Stations.

Table A-5 Primary (Integrated Bus) AViiON Station Device Specifications

Device Name	Unit	3rd Parameter	Example Specification
inen	-	-	inen()
grfx	-	-	grfx()
lp	0-2	-	lp()
duart	-	-	duart()
kbd	-	-	kbd()

Table A-6 Secondary Bus (SCSI) AViiON Station Device Specifications

SCSI Devices

Device Name	Controller	SCSI ID	3rd Parameter	Example Specification
sd	see insc below	0-3	-	sd(insc(),0)
st	see insc below	4-6	file	st(insc(),4)

SCSI Adapter

Device Name	Adapter	SCSI ID	3rd Parameter	Example Specification
1st insc	0	-	-	insc()

Nodes and Device Codes

The following sections give device code and node information for disks, tapes, terminals, and all other devices supported by the DG/UX system.

Physical Disk Nodes

Each physical disk is accessible in either block or character mode. Block-access nodes are in **/dev/pdsk**. Character-access nodes are in **/dev/rpdsk**.

Naming Physical Disks

The following are example entries created in **/dev**:

/dev/pdsk/cied@18(FFFFEE00,0)
/dev/rpdsk/cied@18(FFFFEE00,0)

/dev/pdsk/cied@18(FFFFEE00,1)
/dev/rpdsk/cied@18(FFFFEE00,1)

Logical Disks

Logical disks are accessible in either block or character mode. Therefore, the kernel creates both types of nodes. Block-access nodes are in **/dev/dsk** and character-access nodes are in **/dev/rdsk**. When you use **diskman** or **sysadm** to create a logical disk, a corresponding logical disk node is created. The minor number depends on the order a given disk was added. For instance, the first disk has minor number 0, the second has 1, and so on. If you create a logical disk named **comm**, the nodes in **/dev** will be:

/dev/dsk/comm
/dev/rdsk/comm

Tape Drive Nodes

Tape drives are accessible only in character mode. The character-access nodes are in **/dev/rmt**. Our example system would have **/dev/rmt/st(cisc@28(FFFFF300,4)**. All possible combinations of density and rewind options are created for each unit as follows:

dev/rmt/*diskname@device_code(address)*{**lmh**}[**n**]

Definitions:

st(cisc)	Refers to a SCSI tape with a Ciprico SCSI adapter with default device code and address. The tape is at the default SCSI id.
lmh	low, medium, and high density
n	no rewind on close

Terminal Nodes

Nodes are created in **/dev** from the information in your system file for each terminal line on your terminal controllers. The node names are in the form **tty***xxx*, where *xxx* corresponds to the minor number (line 5 is 05, line 100 is 100, and so on). All terminal nodes are character-access only.

AViiON Workstation tty Assignments

The kernel automatically configures the following ttys on AViiON workstations:

mouse	tty00
duart async	tty01

AViiON System tty Assignments

Depending on whether a cluster controller or a multiplexor (MUX) is present on the system, ttys are automatically configured as follows:

Cluster controller --

The tty lines from 0 to 255 are all assigned no matter how many lines are on the cluster controller. **syac** lines are configured first, then **duart**.

MUX --

Per MUX unit, tty lines are assigned 0-15, 16-31, 32-47, and 48-63. **syac** lines are configured first, then **duart**.

End of Appendix

Appendix B
Directories and Files

This appendix lists the directories and files of the DG/UX system that are of interest to a system administrator. For a list of all directories and files shipped with the DG/UX system, look in **/usr/release/dgux_*.fl** For detailed information on any of the entries here, see the relevant manual reference page.

First in this appendix, we show the **root, var, usr,** and **srv** directories. Next we, show 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). When we showed the DG/UX directory tree in Chapter 2, we denoted a symbolic link by enclosing a directory in parentheses; we'll do the same in this appendix.

Directories in /

The **root** logical disk is mounted on a directory called **/**. We refer to those files on the **root** logical disk as "being in root." Remember that there are physical and logical directories on the **root** logical disk. They are:

(/bin)	Symbolic link to **/usr/bin**. Contains public commands.
(/lib)	Symbolic link to **/usr/lib**. Contains object libraries.
/admin	Home directory for the **sysadm** login.
/dev	Contains device nodes.
/etc	Contains configuration files and system data.
/usr/sbin/init.d	The **init.d** directory contains executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc*N*.d**, where *N* is the appropriate run level. Files are only executed from **/etc/rc*N*.d** directories.

/etc/rc*.d	There are eight of these directories in **/etc**: **rcS.d, rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d,** and **rc6.d**. Each directory contains links to all the shell scripts in **/etc/init.d**. See Chapter 3 for a complete list of all services that can be set for each run level.
/sbin	Contains minimum system commands to get the system up.
/tmp	Used for temporary system files.
/opt	Applications package parent directory.
/local	Contains site-specific files.
/tftpboot	Contains links to client's bootable executable files in **/usr/stand**.
/srv	A mount point for the **srv** logical disk. Contains client and release management files. See "Directories in /srv."
/var	Used for system data files whose size varies as the system runs. See "Directories in /var."

Directories in /var

The **/var** directory contains files that are release dependent, have read and write permissions set, and are dynamically sized.

/var/adm	This directory contains the data collection files for the accounting system. See Chapter 15 for complete information on accounting system files and directories.
/var/Build	Kernel builds by **sysadm newdgux** take place here.
/var/mail	Contains **mail** databases and dynamically sized files.
/var/news	Contains **news** databases and dynamically sized files.
/var/opt	Application package parent directory for dynamically sized files.
/var/spool	Contains spooling files for **lp** and **uucp**.

/var/spool/lp	Contains all of the files and directories of the LP system. See Chapter 11 for complete information on these files and directories.
/var/spool/cron	Contains **cron** databases and dynamically sized files.
/var/spool/uucp	Contains files specific to **uucp**.
/var/preserve	Text editor file save area for sudden program halt.
/var/tmp	User temporary file space.

Directories in /usr

The **usr** logical disk is mounted on the **/usr** directory. Files in this directory are release dependent and read-only. There are physical and logical directories on the **usr** logical disk. They are:

(/usr/spool)	Symbolic link to **/var/spool**.
(/usr/mail)	Symbolic link to **/var/mail**.
(/usr/adm)	Symbolic link to **/var/adm**.
(/usr/news)	Symbolic link to **/var/news**.
(/usr/ucb)	Symbolic link to **/var/ucb**.
(/usr/preserve)	Symbolic link to **/var/preserve**.
/usr/admin	Contains the files, directories, tables, menus, and defaults used by the **sysadm** system administration program.
/usr/bin	Contains user commands.
/usr/catman	Contains the online manual reference pages that users access with the **man(1)** command.
/usr/lib/gcc	Contains the DG/UX GNU C compiler. For information see the Release Notice that comes with the compiler package.
/usr/etc	Contains database and configuration files.
/usr/etc/master.d	Contains master files. These files list devices and kernel parameters for the DG/UX system. This directory may also

	contain master files for other packages that have kernel components, such as TCP/IP and NFS.
/usr/include	Contains include files for system software.
/usr/lib	Contains library routines.
/usr/lib/acct	Contains the C language programs and shell procedures that drive the accounting system. See Chapter 15.
/usr/local	Contains site-specific, read-only files.
/usr/release	Contains media notices, release notices, and system package names.
/usr/opt/*/release	Contains applications packages.
/usr/sbin	Commands used only by an administrator.
(/usr/share)	Symbolic link to **/srv/share.**
/usr/src	Parent directory for source code.
/usr/src/uts/aviion/lb	Contains the kernel libraries which are used to build the kernel image. See Chapter 4. Also see **config**(1M) and **make**(1).
/usr/stand	Contains stand-alone utilities and bootstrap programs.
/usr/tmp	Applications package parent directory.

Directories in /srv

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.
/srv/tftpboot	Contains bootstraps for each release stored here.
/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 an exhaustive look at the DG/UX files. Here, we're trying to highlight the files that you'll be using more often than others. Subsections here are:

- /etc Database Files Maintained via sysadm
- /etc Database Files Maintained Manually
- /sbin Commands
- /usr Files
- /var Files

/etc Database Files Maintained via sysadm

/etc/fstab

The **fstab** file specifies the file system(s) to be mounted by the **/etc/mount** command. The following is a sample entry in **fstab**. Note that it is in ONC/NFS format; we recommend this format even if you are not using ONC/NFS.

```
    /dev/dsk/c24d0  /  dg/ux  rw  1  1
```

The above entry indicates a local file system mount, and the following entry indicates an NFS remote file system mount.

```
   titan:/usr/titan  nfs  rw,hard  0  0
```

The **fstab** format was changed to support ONC/NFS filesystems as well as local file systems. The old style **fstab** entries are also supported. See **fstab(4)** for detailed information.

/etc/gettydefs

The **/etc/gettydefs** file contains information that **/etc/getty** uses to set the speed and terminal settings for a line. The **getty** command accesses the **gettydefs** file with a label (typically the baud rate). The general format of the **gettydefs** file is as follows:

```
label# initial-flags # final-flags #login-prompt #next-label
```

Each line entry in the **gettydefs** file is followed by a blank line. Refer to **gettydefs(4)** for more information. A typical **/etc/gettydefs** file follows:

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
console# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #Console Login:
```

/etc/group

The **/etc/group** file describes each group to the system. An entry is added for each new group. Each entry in the file is one line and consists of four fields, which are separated by a colon (:):

group name:password:group id:login names

See "Procedure 14.6: Add Groups" and **group(4)** for more information. If you have NFS, see **yppasswdd(1M)**.

/etc/inittab

The **/etc/inittab** file contains instructions for the **/etc/init** command. The instructions define the processes that are to be created or terminated for each initialization state. Initialization states are called run levels. By convention, run level S is single-user mode; run level 1 is administrative mode; and run levels 2 and 3 are multi-user modes. Chapter 3 summarizes the various run levels and describes their uses. See **inittab(4)** for more information.

/ettc/passwd

The **/etc/passwd** file identifies each user to the system. Add an entry for each new user. Each entry in the file is one line and consists of seven fields. The fields are separated by colons (:). The fields are:

```
login_name:password:uid:gid:comment:home_directory:program
```

Example:

```
poulet:Rm27oQak1:103:104:L.Q. Poulet:/usr/poulet:/bin/csh
```

See "The User's Environment" in Chapter 14 and **passwd(4)**.

/etc/TIMEZONE

The **/etc/TIMEZONE** file sets the time zone shell variable TZ. The TZ variable in the **TIMEZONE** file is changed by the **sysadm datetime** command. The TZ variable can be redefined on a user (login) basis by setting the variable in the associated **.profile** or **.login**. See Chapter 4.

/etc/dgux.params

This file contains parameters that you can set to control the actions of rc scripts in **/etc/init.d**.

/etc/log

Contains log information on run level changes and daemon activity.

/etc/nfs.params

This file contains parameters for controlling ONC/NFS programs.

/etc/tcpip.params

This file contains the parameters for commands invoked by the **rc** scripts to initialize the network. Chapter 3 describes the **rc** scripts in detail.

/etc/dgux.rclinktab

This data table can be executed by **rc.links** to create or remove links from the **rc#.d** directories to the **/usr/sbin/init.d** directory.

/etc/dgux.prototab

A file listing all DG/UX prototype files in **/etc**.

/etc/dumptab

This file contains the dump table which lists the different media supported by **dump2**(1). It describes the media characteristics for each medium made available to **dump2**. Contains the file formats for incremental dumps.

/etc/inetd.conf

Contains the Internet server configuration database. This is a list of servers that **inetd** invokes when it receives an Internet request over a socket.

/etc Files Maintained Manually

The following files are not maintained by the **sysadm** program.

/etc/login.csh

The default profile for **csh** users is the **/etc/login.csh** file. The default profile for **sh** users is the **/etc/profile** file. The standard (default) environment is established by the instructions in these global profile files. See "The User's Environment" in Chapter 14 for more details.

/etc/devlinktab

This files contains entries used to make short-named links to device nodes with otherwise unwieldy names. An example follows:

```
    /dev/rmt          0          st(insc@99(0,0,0),4)
```

/etc/motd

The **/etc/motd** file contains the message-of-the-day. The message-of-the-day is output by instructions in the **/etc/profile** file after a successful login. This message should be kept short and to the point.

/etc/profile

The default profile for **sh** users is in the **/etc/profile** file. The default for **csh** users is the **/etc/login.csh** file. The standard (default) environment is established by the instructions in these global profile files. See Chapter 14 for examples.

/etc/utmp

The **/etc/utmp** file contains information on the run level of the system. This information is accessed with a **who -a** command.

/etc/wtmp

The **/etc/wtmp** file contains a history of system logins. The owner and group of this file must be **adm**, and the access permissions must be 664. Each time **login** is run this file is updated. As the system is accessed, this file increases in size. Periodically, this file should be cleared or truncated. The command line **>/etc/wtmp** when executed by **root** creates the file with nothing in it. The following command line limits the size of the **/etc/wtmp** file to the last 3600 characters in the file:

```
tail -3600c /etc/wtmp > /tmp/wtmp; mv /tmp/wtmp /etc/wtmp
```

/sbin Commands

The following commands are available in **/sbin**. These are the minimum system administration commands necesary to get a system running.

/sbin/init

Command to change run levels S, 1, 2, 3.

/sbin/sh

The DG/UX **sh**(1) command.

/sbin/su

Command to switch user. For instance **su sysadm** switches you to the **sysadm** login.

/sbin/fsck

Command to run the **fsck**(1M) file system check program.

/sbin/mount

Command to mount a file system on the DG/UX directory tree.

/sbin/umount

Command to unmount a filesystem.

/sbin/chk.fsck

An **rc** check script to run the **fsck** program.

/sbin/shutdown

Command to bring the operating system down to init S. See Chapter 4.

/sbin/halt

Command to halt the operating system to the SCM prompt.

/sbin/setup.d/boot

Set up scripts that must run on the host CPU.

/sbin/setup.d/root

Set up scripts that modify a host's **root** space.

/sbin/rc.init

This program 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
```

/usr Files

/usr/lib/cron/log

A history of all actions taken by **/etc/cron** is recorded in the **/usr/lib/cron/log** file. The **/usr/lib/cron/log** file should be periodically truncated to keep the size of the file within a reasonable limit. The following command line limits the size of the log file to the last 100 lines in the file:

```
tail -100 /usr/lib/cron/log > /tmp/log; mv /tmp/log /usr/lib/cron/log
```

/usr/sbin/init.d

This directory contains the the systems's check scripts and **rc** scripts.

/usr/sbin/setup.d/usr

This directory contains set up scripts that modify a host's **usr** space. Setup scripts might include those for TCP/IP and NFS.

/usr/src/uts/aviion/cf/system.*.proto

This file contains your custom version of the devices and configuration parameters listed in **/usr/etc/master.d/dgux**. For configuration, the **config(1M)** program runs on the **system** file and produces program code in **conf.c**. Prototype system files are shipped with software packages with kernel content. Prototype files have names of the form **system.*.proto**.

/var Files

/var/adm/sulog

The **/var/adm/sulog** file contains a history of superuser (**su(1)**) command usage. As a security measure, this file should not be readable by others. The **/var/adm/sulog** file should be periodically truncated to keep the size of the file within a reasonable limit. The following command lines limit the size of the log file to the last 100 lines in the file:

```
tail -100 /usr/adm/sulog > /tmp/sulog
```

```
mv /tmp/sulog /usr/adm/sulog
```

A typical **/usr/adm/sulog** file follows:

```
SU 08/18 16:16 + console smitht-root
SU 08/18 23:45 + tty00 jones-root
SU 08/19 11:53 + console smitht-sysadm
SU 08/19 15:25 + console root-sysadm
SU 08/19 23:45 + tty00 root-uucp
```

/var/spool/cron/crontabs

The **cron** function is useful for doing recurring and habitual system administration tasks. The **/var/spool/cron/crontabs** directory contains crontab files for **adm**, **root**, and **sys** logins. Providing their lognames are in the **/usr/lib/cron/cron.allow** file, users can establish their own **crontabs** file using the **crontab** command. If the **cron.allow** file does not exist, the **/usr/lib/cron/cron.deny** file is checked to determine if the user is denied the use of the **crontab** command.

As **root** or **sysadm**, you can either use the **crontab(1)** command or edit the appropriate file under **/var/spool/cron/crontabs** to make the desired entries. Revisions to the file take effect at the next reboot. Refer to **crontab(1)** for additional information.

End of Appendix

Appendix C
Error Messages

This appendix contains error messages for the LP system, UUCP, and for system calls (errors that return *errno* error numbers). For information on changing system call error message files, see **ermes_editor(1M)**. PANIC error messages are shipped online and are located in **/usr/release/dgux_4.10.panic**. You should print out a paper copy of the PANIC error messages and and keep it handy in case of system failure.

LP System Error Messages

This section provides a description of the error messages that are associated with LP commands. The following variables are used in the LP error messages:

file(s)	Indicates the file or files that are to be printed.
dest	Indicates the name of the destination printer.
printer id	Indicates the request identification number of the printout. For example, *newlp-10* is the printer name followed by the request identification number.
printer name	Indicates the name of the printer.
program name	Indicates the program name that was executed.
user	Indicates the user who requested the printout.

Following each message is an explanation of the probable cause of the error and the corrective action to take. If you are not able to correct all the error conditions you encounter, call your service representative for assistance.

LP Error Message	Description/Action
acceptance status of destination "printer name" unknown	Use the **sysadm acceptlp** command to enable the printer so that it will accept requests.
dest is an illegal destination name	The *dest* you used is not a valid destination name. Use the **lpstat -p** command to list valid destination names.
file is a directory	The file name you typed is a directory and cannot be printed.
xx is not a request id or a printer	The argument you used with the **cancel** command is not a valid request identification number or a printer name. Use the **lpstat -t** command to give you all the printers and requests waiting to get printed.
xx is not a request id	The request identification number you used with the **lpmove** or **sysadm movelp** command is not a valid request identification number. To find out what requests are valid, use the **lpstat -u** command.
xx not a request id or a destination	You used an invalid request identification number or destination. To find out what is valid, use the **lpstat -t** command.
dest not accepting requests since *date*	The printer you are trying to use is in reject mode.
Can't access FIFO	The named pipe file **/var/spool/lp/FIFO** is incorrect. The mode should be 600.
can't access file xx	The mode could be wrong on your directory or the file that you are trying to access.
can't create class "xx"-it is an existing printer name	The class name you are trying to use has already been given to a printer. You will have to use another name or remove the printer to use the class name.
can't create new acceptance status file	The mode may be wrong on the **/var/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."

LP Error Message	Description/Action
can't create new class file	The mode may be wrong on the **/var/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new interface program	The mode may be wrong on the **/var/spool/lp/interface** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new member file	The mode may be wrong on the **/var/spool/lp/member** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new printer status file	The mode may be wrong on the **/var/spool/lp/pstatus**. It should be 644 with the owner "lp" and the group "bin."
can't create new request directory	The mode may be wrong on the **/var/spool/lp/request** directory. It should be 755 with the owner "lp" and the group "bin."
can't create printer "printer name" -- it is an existing class name	The printer name you are trying to use has already been used as a class name. You will have to assign another name for the printer. Use **sysadm modlp**.
can't move request printer id	*Printer id* is the request identification number that cannot be moved. You will probably have to change the modes on the files and directories in **/var/spool/lp/request**. Also, you will have to manually move the request from the disabled printer directory to the new destination after you shut down the LP scheduler.
can't create new output queue	The mode on the file **/var/spool/lp/seqfile** is incorrect. It should be 644, and the mode on the directory should be 755. The owner should be "lp," and the group should be "bin."
can't create new sequence number file	The mode on the file **/var/spool/lp/seqfile** is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin."

LP Error Message	Description/Action
can't create request file "xx"	The mode on the file **/var/spool/lp/request/***printer name*/r-id is incorrect. *Printer name* is the name of the printer such as dqp10, and r-id is the request identification number. The mode of the file should be 444, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin."
can't fork	You either have several processes running and are not allowed to run anymore, or the system has all the processes running that it can handle. You will have to rerun this command later.
can't lock acceptance status	This is a temporary file in **/var/spool/lp** that prevents more than one "lp" request from being taken at any given instant. You will have to rerun this command later.
can't lock output queue	The file **/var/spool/lp/QSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You will have to rerun this command later.
can't lock printer status	The temporary file **/var/spool/lp/PSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You will have to rerun this command later.
can't lock sequence number file	The file **/var/spool/lp/SEQLOCK** prevents more than one "lp" request from getting the next printer id (request identification) number at a time. Try this command again later.
can't open class file	The **lp** program is trying to access the list of classes for printers. One reason it may not be able to open the class file is that the system could have the maximum number of files open that are allowed at any time. Try the command again later.
can't open member file	The **lp** program is trying to access the list of members in the directory **/var/spool/lp/member**. The system could have the maximum number of files open that are allowed at any time. Try this command again later.

LP Error Message	Description/Action
can't open xx file in class directory	One possibility why file *xx* cannot be opened is that the mode on the file or directory is incorrect. The file mode should be 644, and the directory mode should be 755. Another possibility is that the system has the maximum number of files open that are allowed at any time. The latter problem can be corrected by trying the command again later.
can't open xx	You cannot print on printer *xx* because the mode is incorrect on **/dev/tty**. The mode should be 622.
can't open FIFO	The mode on the named pipe file **/var/spool/lp/FIFO** may be incorrect. It should be 600. Or, the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by trying the command again later.
can't open default destination file	Check the mode on the file **/var/spool/lp/default**. The mode should be 644. If the mode is okay, it could be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open file filename	The *filename* was incorrectly typed or you don't have the correct modes set. The mode should be at least 400 if you are the owner.
can't open output queue file	Check the mode on the file **/var/spool/lp/outputq**. It should be 644. This error message could also be generated if the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open printer status file	The mode on the file **/var/spool/lp/pstatus** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open request directory directory name	The mode on the directory **/var/spool/lp/request** is incorrect. The mode should be 655. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.

LP Error Message	**Description/Action**
can't open request file xx	The mode on the file **/var/spool/lp/member/request/***xx* is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command again later.
can't open system default destination file	The mode on the file **/var/spool/lp/default** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open temporary output queue	The mode on the file **/var/spool/lp/outputq** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't proceed -- scheduler running	Many of the **lpadmin** command options cannot be executed while the scheduler is running. Stop the scheduler using the **lpshut** command and then try invoking the command again.
can't read current directory	The **lp** and **lpadmin** commands cannot read the directory containing the file to be printed. The directory name may be incorrect or you do not have read permission on that directory.
can't remove printer	The mode may be wrong on the **/var/spool/lp/member** directory. It should be 755, and the files in that directory should be 644. Both the directory and the files should be owned by "lp," and the group should be "bin."
can't remove request directory	The mode may be wrong on the **/var/spool/lp/request** directory. It should be 755 and should be owned by "lp," and the group should be "bin." The directory may still have pending requests to be printed which will have to be removed before the directory can be removed.
can't set user ID to LP Administrator's user ID	The **lpsched** and **lpadmin** commands can only be used when you are logged in as root.

LP Error Message	Description/Action
can't write to xx	The **lpadmin** command cannot write to device *xx*. The mode is probably wrong on the /dev/ttyxx file. It should be 622 and owned by "lp."
cannot create temp file filename	The system may be out of free space on the **/usr** file system. Use the command **df /usr** to determine the number of free blocks. Several hundred blocks are required to insure that the system will perform correctly.
class "xx" has disappeared!	Class *xx* was probably removed since the scheduler was started. The system may be out of free space on the /usr file system. Use the command **df /usr** to find out. Use the **lpshut** or **sysadm stoplp** command to stop the scheduler and restore the class from a backup.
class "xx" non-existent	The class *xx* may have been removed because the system is out of free space on the **/usr** file system. Use the command **df /usr** to find out how much free space is available. The class will probably have to be restored from a backup.
class directory has disappeared!	The **/var/spool/lp/class** directory has been removed. The system may be out of free space on **/usr**; use the **df /usr** command to find out. The class directory contains all the data for each printer class. To restore this directory, get these files and directory from a backup.
corrupted member file	The **/var/spool/lp/member** directory has a corrupted file in it. You should restore the directory from backup.
default destination "dest" non-existent	Either the default destination is not assigned or the printer *dest* has been removed. Use **lpadmin** or **sysadm defaultlp** to set up a default destination.
destination "dest" has disappeared!	A destination printer, *dest* has been removed since **lpsched** was started. Use **sysadm dellp** to remove the printer.
destination "printer name" is no longer accepting requests	The printer is not accepting requests because the **reject** or **sysadm rejectlp** command has been invoked. Use **sysadm acceptlp** or **accept** to make the printer accept requests.

LP Error Message	Description/Action
destination dest non-existent	The destination printer you specified as an argument to the **accept** or **lpadmin** command is not a valid destination name, or it has been removed since the scheduler was started.
destination "printer name" was already accepting requests	The destination printer was previously "enabled." Once a printer is accepting requests, issuing any more **accept** commands to it are ignored.
destination "printer name" was already not accepting requests	A **reject** command was already sent to the printer. Use the **accept** command to allow the printer to start accepting requests again.
destination printer name is not accepting requests move in progress ...	The printer has been disabled by the **reject** command, and requests are being moved from the disabled printer to another printer. The printer can be enabled again by the **accept** command.
destinations are identical	When using the **lpmove** command, you need to specify a printer to move the print requests from and a different printer to move the requests to.
disabled by scheduler: login terminal	The login terminal has been disabled by the LP scheduler. The printer can be reenabled by using the **enable** or **sysadm enablelp** command.
error in printer request printer id	*Printer id* is the actual request identification number. The error was most likely due to an error in the printer. Check the printer, and reset it if needed.
illegal keyletter "xx"	An invalid option, *xx*, was used. See the manual page for the correct options.
keyletters "-xx" and "-yy" are contradictory	This combination of options to the **lpadmin** program cannot be used together.
keyletter "xx" requires a value	The option *xx* requires an argument. For example, in the command line **lpadmin -m** *model* the argument to the **-m** option is the name of a model interface program.

LP Error Message	Description/Action
keyletters -e, -i, and -m are mutually exclusive	These options to the **lpadmin** command cannot be used together. Refer to the manual page for more information.
LP Administrator not in password file	You must have an entry in the **/etc/passwd** file for lp, and you must belong to the group bin.
model "xx" non-existent	The name that you are using for a model interface program is not a valid one. A list of valid models is in the **/var/spool/lp/model** directory.
new printers require -v and either -e, -i, or -m	A printer must have an interface program, and this is specified by **-e**, **-i**, or **-m** options. The **-v** option specifies the device file for the printer. For more information on these options, refer to the **lpadmin** manual page.
no destinations specified	There are no destination printers specified. Use the **lpadmin** command to set one up.
no printers specified	There are no printers specified. Use the **lpadmin** command to set one up.
non-existent printer xx in PSTATUS	A printer with the name *xx* is in the **/var/spool/lp/pstatus** file but no longer exists. The printer should be removed using the **lpadmin** command.
non-existent printer in class xx	The printer that you are trying to address in class *xx* has been removed from that class.
out of memory	Implies the system is in trouble. The message implies that there is not enough memory to contain the text to be printed.
printer "printer name" already in class "xx"	The printer you are trying to move to class *xx* is already in that class. You cannot move a printer to a class that it is already in.

LP Error Message	Description/Action
printer "printer name" has disappeared!	The printer has been removed, and the **enable** command cannot find it. The printer was most likely removed since the machine was rebooted or since the scheduler was started.
printer "printer name" non-existent	*Printer name* is the name of a printer that has been removed since the scheduler has been started. You must use the **lpadmin -x***printer name* command.
printer status entry for "printer name" has disappeared	The **/var/spool/lp/pstatus** file must have been corrupted. You will have to resubmit the printer request.
printer "printer name" was not busy	The printer is not printing a request at this time. Either the request you wanted to cancel is finished printing or you have specified the wrong printer.
request "printer id" non-existent	You are attempting to cancel a request that does not exist. You may have given the wrong printer name or wrong request id number or the request may have finished printing.
request not accepted	The request was not accepted by **lp**. The scheduler may not be running. Use the **lpstat -t** command to find out more information.
requests still queued for "printer name" use lpmove	*Printer name* is the printer that still has requests waiting to get printed. You need to use the **lpmove** command to get those requests moved to another printer.
scheduler is still running -- can't proceed	You cannot perform this command while the scheduler is running. You will have to use the **lpshut** command first.
spool directory non-existent	The directory **/var/spool** has been removed. You will have to use the **mkdir** command to restore the directory. This has probably removed some of the necessary LP files. You may have to reinstall the LP commands.
standard input is empty	You specified an invalid file name either by incorrectly typing a name or by specifying a nonexistent file. Nothing will be printed on the printers from this request.

LP Error Message	Description/Action
this command for use only by LP Administrators	This command is restricted to someone logged in as root or lp.
too many options for interface program	The **lp** command called the appropriate interface program with too many arguments. For more information on the options and arguments that can be used with the **lp** command, refer to the **lp** manual page.
unknown keyletter "xx"	An invalid option was supplied to the **lp** or **lpadmin** command.
unknown option "xx"	This message is displayed in response to an invalid option supplied to the **disable**, **lpstat**, or **reject** commands. Refer to Chapter 10 of this manual.
usage: disable [-c] [-r[reason]] printer	The syntax for the **disable** command is not correct. The valid options are: **-c** to cancel the currently printing request, and **-r** followed by the reason that you are disabling the printer.
usage: reject [-r[reason]] dest ...	The syntax for the **reject** command is not correct. The proper format is to specify the reason why the printer is not taking any more print requests and to identify the destination printer.
usage: accept dest	The syntax for the **accept** command is not correct. You did not specify what printer should accept requests.
usage: enable printer	The syntax for the **enable** program is to specify a destination printer.
usage: cancel id .... printer	The syntax for the **cancel** command is not correct. The proper format is to specify the request identification number or the printer name.

LP Error Message	Description/Action
usages: lpadmin -pprinter [-vdevice] [-cclass] [-rclass] [-eprinter | -iinterface | -mmodel] [-h | -l] -or- lpadmin -d[destination] -or- lpadmin -xdestination	The correct syntax for the **lpadmin** command is to specify at least one of the options mentioned above.
your printer request id was canceled by user	The printer request did not finish printing because another *user* canceled it. Typically, you will get this message in your mail. One reason a person may cancel a request other than their own is because the request is not printing correctly.

UUCP Error Messages

This section lists two types of error 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 process is aborted, ASSERT error messages are recorded in **/var/spool/uucp/.Admin/errors**. These messages include the filename, sccsid, line number, and the text listed below. In most cases, these errors are the result of file system problems.

Assert Error Message	**Description/Action**
BAD LINE	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
BAD SPEED	A bad line speed appears in the **Devices/Systems** files (Class field).
BAD LOGIN_UID	The uid cannot be found in the **/etc/passwd** file. The file system is in trouble, or the **/etc/passwd** file is inconsistent.
BAD UID	Same as previous.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T CHDIR	A chdir() call failed.
CAN'T CHMOD	A chmod() call failed.
CAN'T CREATE	A create() call failed.
CAN'T CLOSE	A close() or fclose() call failed.
CAN'T FORK	An attempt to **fork** and **exec** failed. The current job should not be lost, but will be attempted later (**uuxqt**). No action need be taken.
CAN'T LINK	A link() call failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed.
CAN'T OPEN	An open() or fopen() failed.
CAN'T READ	A read(), fgets(), etc. failed.
CAN'T STAT	A stat() call failed.
CAN'T WRITE	A write(), fwrite(), fprint(), etc. failed.
CAN'T UNLINK	An unlink() call failed.
WRONG ROLE	This is an internal logic problem.

Assert Error Message	**Description/Action**
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.
ULIMIT TOO SMALL	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
SYSLST OVERFLOW	An internal table in **gename.c** overflowed. A big/strange request was attempted.
TOO MANY FILES	Same as previous.
RETURN FROM fixline ioctl	An ioctl, which should never fail, failed. There is a system driver problem.
PERMISSIONS file: BAD OPTION	There is a bad line or option in the **Permissions** file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
SYSTAT OPEN FAIL	There is a problem with the modes of **/usr/lib/uucp/.Status**, or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem! Contact your Data General Representative.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were suppose to be checked before this process was attempted.

STATUS Messages

Status messages are stored in the **/var/spool/uucp/.Status** directory. This directory contains a separate file for each remote machine that your system attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common messages that may appear in these files.

Status Message	Description/Action
ASSERT ERROR	An ASSERT error occurred. Check the **/var/spool/uucp/.Admin/errors** file for the error message.
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the **Permissions** file.
CALLBACK REQUIRED	The called machine requires that it calls your DG/UX System.
CALLER SCRIPT FAILED	This is usually the same as "DIAL FAILED." However, if it occurs often, suspect the caller script in the **dialers** file. Use **uutry** to check.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DEVICE FAILED	The open of the device failed.
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer-Token-Pairs script.

Status Message	Description/Action
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
OK	Things are OK.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your computer in its **Systems** file.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your computer. They could be trying to call your machine. If they have an older version of UUCP, the process that was talking to your machine may have failed leaving the LCK file. Check to see if the process that has a LCK file is hung.
REMOTE REJECT AFTER LOGIN	The login used by your computer to login does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your computer for an unknown reason. The remote machine may not be running a standard version of UUCP.
STARTUP FAILED	Login succeeded, but initial handshake failed.
SYSTEM NOT IN Systems	The system is not in the **Systems** file.
TALKING	Self explanatory.
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.

Errno Error Messages

Errno stands for error number. When an error occurs that originates in the misuse of a system call, an *errno* error message appears on the screen. A programmer defines these messages in **<errno.h>**. The following is an example of such a message.

Let's say you try to copy **file1** in a directory where you do not have write permissions. You type:

```
$   cp file1  my_file
```

The system responds:

```
$      1: not owner
```

So, you are not allowed to copy the file and you get an *errno* error message. When an *errno* error is displayed at your terminal and you are unsure of the meaning, check the following list: It is a complete general reference list of all error numbers that system calls might return.

Errno Error Message	Description/Action
1 EPERM	Not owner. This error usually indicates an attempt to modify a file in some way forbidden except to its owner or to the superuser. It also indicates attempts by ordinary users to do things allowed only to the superuser.
2 ENOENT	No such file or directory. This error occurs when you try to use a pathname that is too long, refer to a file that doesn't exist, or use a pathname that includes an invalid directory name (e.g., the directory doesn't exist).
3 ESRCH	No such process. No process can be found corresponding to that specified by **pid** in **kill** or **ptrace**.
4 EINTR	Interrupted system call. An asynchronous signal (such as interrupt or quit), which the user has elected to catch, occurred during a system call. If execution resumes after processing the signal, the interrupted system call will seem to return this error condition.
5 EIO	I/O error. Some physical I/O error has occurred. This error may occur on a call following the one to which it actually applies.
6 ENXIO	No such device or address. I/O on a special file refers to a subdevice that does not exist, or that extends beyond the limits of the device. It may also occur when a device is not on-line or no disk pack is loaded on a drive.
7 E2BIG	Argument list too long. An argument list longer than 5,120 bytes is presented to a member of the **exec** family.
8 ENOEXEC	Exec format error. A request is made to execute a file which, although it has the appropriate permissions, does not start with a valid magic number (see **a.out (4)**).
9 EBADF	Bad file number. Occurs under any of three conditions: a file descriptor refers to no open file; a read request is made to a file that is open only for writing; a write request is made to a file that is open only for reading.

Errno Error Message	Description/Action
10 ECHILD	No child processes. A **wait** was executed by a process that had no existing or unwaited-for child processes.
11 EAGAIN	No more processes. A **fork** failed because the system's process table was full or the user was not allowed to create any more processes.
12 ENOMEM	Not enough space. During a **brk** or **sbrk**, a program asks for more space than the system can supply. This is not a temporary condition; the maximum space size is a system parameter. The parameter is a program size switch that is set for the linker. The usual size is 1M bytes, maximum is 512M bytes.
13 EACCES	Permission denied. You tried to access a file in a way forbidden by the protection system.
14 EFAULT	Bad address. The system encountered a hardware fault in attempting to use an argument of a system call.
15 ENOTBLK	Block device required. A non-block file was mentioned where a block device was required, e.g., in **mount**.
16 EBUSY	Device or resource busy. You tried to mount a device that was already mounted or to dismount a device on which there is an active file (open file, current directory, mounted-on file, active text segment). This error will also occur if you try to enable accounting when it is already enabled, or if device or resource requested is currently unavailable.
17 EEXIST	File exists. An existing file was mentioned in an inappropriate context; e.g., **link**.
18 EXDEV	Cross-device link. You tried to link to a file on another device.
19 ENODEV	No such device. You tried to apply an inappropriate system call to a device; e.g., read a write-only device.
20 ENOTDIR	Not a directory. You gave a non-directory reference where a directory reference is required.

Errno Error Message	**Description/Action**
21 EISDIR	This is a directory. An attempt was made to write on a directory.
22 EINVAL	Invalid argument. Some invalid argument; e.g., dismounting a non-mounted device, mentioning an undefined signal in **signal** or **kill** , reading or writing a file for which **lseek** has generated a negative pointer. Also set by the math functions described in the (3M) reference pages.
23 ENFILE	File table overflow. The system file table is full, and no more **opens** can be accepted now.
24 EMFILE	Too many open files. No process may have more than 20 file descriptors open at a time.
25 ENOTTY	Not a character device. You tried to **ioctl(2)** a file that is not a special character device.
26 ETXTBSY	Text file busy. You tried to execute a pure-procedure program that is currently open for writing. Also occurs if you try to open for writing a pure-procedure program that is being executed.
27 EFBIG	File too large. A file exceeded the maximum file size (1,082,201,088 bytes) or see **ulimit(2)**.
28 ENOSPC	No space left on device. During a **write** to an ordinary file, there is no free space left on the device.
29 ESPIPE	Illegal seek. An **lseek** was issued to a pipe or socket.
30 EROFS	Read-only file system. You tried to modify a file or directory on a device mounted read-only.
31 EMLINK	Too many links. You tried to make more than the maximum number of links (1000) to a file.

Errno Error Message	Description/Action
32 EPIPE	Broken pipe. A write on a pipe for which there is no process to read the data. This condition normally generates a signal; the error is returned if the signal is ignored.
33 EDOM	Math argument. The argument of a function in the math package (3M) is out of the domain of the function.
34 ERANGE	Result too large. The value of a function in the math package (3M) is not representable within machine precision.
35 ENOMSG	No message of desired type. An attempt was made to receive a message of a type that does not exist on the specified message queue; see **msgop(2)**.
36 EIDRM	Identifier removed. This error is returned to processes that resume execution due to the removal of an identifier from the file system's name space. See **msgctl(2)**, **semctl(2)**, and **shmctl(2)**.
37 ECHRNG	Channel number out of range.
38 EL2NSYNC	Level 2 not synchronized.
39 EL3HLT	Level 3 halted.
40 EL3RST	Level 3 reset.
41 ELNRNG	Link number out of range.
42 EUNATCH	Protocol driver not attached.
43 ENOCSI	No CSI structure available.
44 EL2HLT	Level 2 halted.
45 EWOULDBLOCK	Operation would block. An operation that would cause a process to block was attempted on a object in non-blocking mode. See **ioctl(2)**.

Errno Error Message	**Description/Action**
46 EINPROGRESS	Operation now in progress. An operation that takes a long time to complete (such as a **connect(2)**) was attempted on a non-blocking object. See **ioctl(2)**.
47 EALREADY	Operation already in progress.
48 ENOTSOCK	Socket operation on non-socket. Self-explanatory.
49 EDESTADDRREQ	Destination address required. A required address was omitted from an operation on a socket.
50 EMSGSIZE	Message too long. A message sent on a socket was larger than the internal message buffer.
51 EPROTOTYPE	Protocol wrong type for socket. You specified a protocol that does not support the semantics of the socket type requested. For example, you cannot use the ARPA Internet UDP protocol with type SOCK_STREAM.
52 ENOPROTOOPT	Bad protocol option. You specified a bad option in a **getsockopt(2)** or **setsockopt(2)** call.
53 EPROTONOSUPPORT	Protocol not supported. The protocol has not been configured into the system or no implementation for it exists.
54 ESOCKTNOSUPPORT	Socket type not supported. The support for the socket type has not been configured into the system or no implementation for it exists.
55 EOPNOTSUPP	Operation not supported. For example, trying to **accept** a connection on a datagram socket.
56 EPFNOSUPPORT	Protocol family not supported. The protocol family has not been configured into the system or no implementation for it exists.
57 EAFNOSUPPORT	Address family not supported by protocol family. You used an address incompatible with the requested protocol. For example, you can't always use PUP Internet addresses with ARPA Internet protocols.

Errno Error Message	**Description/Action**
58 EADDRINUSE	Address already in use. Only one usage of each address is normally permitted.
59 EADDRNOTAVAIL	Can't assign requested address. This error usually results from an attempt to create a socket with an address not on this machine.
60 ENETDOWN	Network is down. A socket operation encountered a dead network.
61 ENETUNREACH	Network is unreachable. A socket operation was attempted to an unreachable network.
62 ENETRESET	Network dropped connection on reset. The host you were connected to crashed and rebooted.
63 ECONNABORTED	Software caused connection abort. A connection abort was caused internal to your host machine.
64 ECONNRESET	Connection reset by peer. A connection was forcibly closed by a peer. This normally results from the peer executing a **shutdown(2)** call.
65 ENOBUFS	No buffer space available. An operation on a socket or pipe was not performed because the system lacked sufficient buffer space.
66 EISCONN	Socket is already connected. A **connect** request was made on an already connected socket; or, a **sendto** or **sendmsg** request on a connected socket specified a destination other than the connected party.
67 ENOTCONN	Socket is not connected. An request to send or receive data was disallowed because the socket was not connected.
68 ESHUTDOWN	Can't send after socket shutdown. A request to send data was disallowed because the socket had already been shut down with a previous **shutdown(2)** call.
69 ETOOMANYREFS	Too many references; can't splice.

Errno Error Message	**Description/Action**
70 ETIMEDOUT	Connection timed out. A **connect** request failed because the connected party did not properly respond after a period of time. (The timeout period depends on the communication protocol.)
71 ECONNREFUSED	Connection refused. No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host.
72 ELOOP	Not used in DGUX.
73 ENAMETOOLONG	File name too long.
74 EHOSTDOWN	Host is down.
75 EHOSTUNREACH	No route to host.
76 ENOTEMPTY	Directory not empty.
77 EPROCLIM	Not used in DG/UX.
78 EUSERS	Too many users.
79 EDQUOT	Disk quota exceeded.
80 EDEADLOCK	Deadlock in lockf.

End of Appendix

Appendix D
File System Checking: fsck

The file system check program, **fsck(1)**, is a multipass file system check and repair program. Each file system pass invokes a different phase of the **fsck** program. You must use this program when when you are bringing up your system after an abnormal shutdown such as a power outage or system crash.

The **fsck** program checks, in order, blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region (DAR) information, including the free-block bitmap, the free-inode list, and summary counts. The program reports any inconsistencies. It is your option to fix or ignore them.

This appendix:

- Discusses the normal updating of the file system.
- Discusses the possible causes of file system corruption.
- Presents the corrective actions taken by **fsck**. It describes both the program and the interaction between the program and the system administrator.
- Contains the **fsck** error conditions, giving their meanings, possible responses to them, and related error conditions.

File System Update

Every time a file is modified, the DG/UX operating system performs a series of file system updates. When written to disk, these updates yield a consistent file system.

There are five types of file system updates. These involve the (1) superblock, (2) inodes, (3) index (indirect) blocks, (4) data blocks (directories and other files), and (5) 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. **Fsck** will mark a file system mountable only when it is consistent.

A file system is marked mountable when it is created. It is marked unmountable whenever it is mounted, and does not become mountable again until it is either unmounted or cleanly checked. 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.

Fsck lets you check a file system for structural integrity by performing consistency checks on redundant data. The redundant data is either read from the file system or computed from other known values. When **fsck** reports an inconsistency, it asks if the inconsistency is to be corrected by repairing or deleting the corrupted item. You can can accept or reject this request. In the following example, we invoke **fsck**. The program finds an incorrect link count in Phase 4 and asks if it should fix the problem. We respond with **y** to the query:

```
# 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).

#
```

The **fsck** program corrected the link count of inode 67.

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. These 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.

Free-Block Bitmap

Each DAR contains a bitmap representing all the blocks in the DAR. **Fsck** compares that information with its own map of allocated blocks.

Free-Inode List

Each DAR contains a link list of free inodes in that DAR. **Fsck** traverses that list to ensure that all free inodes from that DAR are in the list, and that no allocated inodes are in it.

Summary Counts

The superblock and each DAR contain several counts: the number of used inodes, the number of used blocks, the number of directories. **Fsck** 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. **Fsck** checks for inconsistencies involving format and type, link count, duplicate blocks, and inode size.

Format and Type

Each inode contains mode information. This information describes the type of the inode. Inodes may be one of eight types: regular, directory, control point directory, special block, symbolic link, special character, FIFO, or socket. Any other type is illegal.

Link Count

Each inode contains a count of the directory entries linked to the inode. **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.

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 this is the case, **fsck** may link the disconnected file to the **lost+found** directory.

If the stored and actual link counts are nonzero and unequal, **fsck** may replace the link count on the disk by the actual link count. When this situation arises, a directory entry may have been added or removed without the inode being updated.

Duplicate Blocks

Each inode contains a list and sometimes pointers to lists (index blocks) of all the blocks claimed by the inode. **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.

Size Checks

Each inode contains a size field. This field's size indicates the number of bytes in the file associated with the inode. **Fsck** 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. **Fsck** also checks for directory corruption, where conflicting information is found within the directory entries.

Fsck can also perform a check of the size field of an inode. **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.

Control Point Directories

The root inode of a file system is a special type of directory known as a *control point directory*. A control point directory is like an ordinary directory except that it has resource limits associated with it, for inodes and for data blocks. The total resources consumed by the control point directory and all its descendants (to which it is the *space parent*) may not exceed the limits.

Index Blocks

Index blocks (also known as indirect blocks) are owned by an inode. Therefore, inconsistencies in an index block directly affect the inode that owns the block. **Fsck** can check inconsistencies involving blocks already claimed by another inode and block numbers outside the range of the file system.

Data Blocks

There are two types of data blocks: plain data blocks and directory data blocks. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries. **Fsck** does not try to check a plain data block.

Fsck can check each directory data block for:

- bad self-identification information
- directory entries for unallocated inodes
- directory entries for inodes which do not exist in the file system
- directories that are disconnected from the file system

If a directory entry inode number points to an unallocated inode, **fsck** may remove that directory entry. This condition usually occurs when the data blocks containing the directory entries are modified and written out, but the inode is not written out.

If a directory entry inode number is pointing to a non-existent inode, **fsck** may remove that directory entry. This condition occurs if bad data is written into a directory data block.

Fsck checks that all directories are linked into the file system; i.e., they have a parent directory pointing to them (except for the root). If **fsck** finds unlinked directories, it links the directory back into the file system in the **lost+found** directory. When inodes are being written to the file system without the corresponding directory data blocks being written to the file system, the directories are not linked into the file system.

Invoking the fsck Program

Fsck can be invoked in four ways:

Startup script This is the most common way of invoking **fsck**. When you are in multi-user mode bringing up the system with the **init** command, you can automatically execute **fsck** from within your startup script.

Command line From the command line, you type:
fsck *[options] [filesystem_names]*.

Initialization This is the version of **fsck** that is built into the operating system kernel and is automatically run over the root file system when you boot your system.

Stand-alone This is the least common method of invoking **fsck**. You will use the stand-alone version of **fsck** via the stand-alone **diskman** menu. See Chapter 6 of this manual.

Options to fsck

All options are represented by single-character flags; options must begin with a hyphen. All options except for **-t** are Boolean flags, and may thus be combined: **fsck -pxD**, for example.

The following options are interpreted by **fsck**:

-p Detect all possible inconsistencies, but correct only those inconsistencies that may be expected to occur from an abnormal system halt. For each corrected inconsistency, one or more lines will be printed identifying the file system and the nature of the correction. Any other inconsistencies will cause the check of that file system to fail. The following 15 inconsistencies (and only those listed) will be corrected

for the specified file systems:

1) An inode has an incorrect count of the blocks it uses. The count is corrected.

2) An inode is partially truncated. This can occur if the system is abnormally halted while a file is being truncated, leaving it 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 filename 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 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**. This option should be used with great caution, since it could lead to irreversible changes to the filesystem.

-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 File systems are examined before being checked. If a file system is marked mountable in its superblock, then it is not checked.

-s Ignore the actual free-block bitmap and unconditionally reconstruct a new one.

-S Conditionally reconstruct the free-block bitmap. It is reconstructed if and only if the file system is consistent. This option also forces a "no" response to all questions.

-t Use the specified scratch file for temporary storage if **fsck** cannot obtain enough memory. The scratch file's name must be the next argument after **-t**.

-D Directories are checked for bad blocks.

-f Fast check: blocks and sizes are checked; the free block bitmap is reconstructed if necessary.

The following options are mutually exclusive, and use of more than one per invocation is not allowed: **-y, -n, -p, -q, -S**.

Arguments

The file system(s) to be checked may be specified either implicitly or explicitly. If no arguments are given, the file systems to be checked may be found in one of two special files: **/etc/checklist** and **/etc/fstab**. If **/etc/checklist** exists, then every entry in it is checked in order. If **/etc/checklist** does not exist, but **/etc/fstab** does, then all the file systems listed with a non-zero pass number and a "rw" or "ro" mounting status are checked. If the **-p** option was specified, the checking occurs in order of pass number, with those file systems of equal pass number being checked in parallel with each other. Otherwise, checking occurs in order of appearance in **fstab**.

If arguments are specified (the rest of the command line after the option flags), those file systems, and only those file systems, are checked sequentially in the order given.

File systems may be specified as arguments to **fsck** in one of two ways: by the special device file (in **/dev/dsk** or **/dev/rdsk**) containing the file system; or by the directory that **/etc/fstab** indicates will serve as the mount point for the file system.

Checking

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; **Fsck** either exits or proceeds to the next specified file system. When an inconsistency is discovered with the **-p** option, and that error is one of those listed under **-p**, the inconsistency is fixed without operator approval. Any other discoveries of inconsistencies require the operator to make a decision. The **fsck** program prompts with its recommended action. If you answer **yes**, then **fsck** takes the recommended action. In no case will any damaging action be taken without approval. Note, however, that advance approval or disapproval may be given by invoking **fsck** with the **-y** and **-n** options, respectively.

The **fsck** program 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 filenames: "/" 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 filename lengths.
- Extraneous hard links to directories (including cycles in file system name space).
- Extraneous hard links to Symbolic Link files.
- Directory entries to invalid inodes.
- Directory entries to unallocated inodes.
- Files with incorrect space parent.
- Unconnected files or directories.
- Bad or missing **lost+found** directories.
- Bad **lost+found** directory entries.
- Root or **lost+found** directories needing expansion.
- Files with incorrect link counts.
- Incorrect resource allocation counts in control point directories.
- Bad format in DAR blocks.
- Invalid contents in reserved area of DAR blocks.
- Incorrect free-block bitmaps in DARs.
- Incorrect or incomplete free-inode lists in DARs.
- Incorrect DAR summary counts: blocks used, inodes used, directories used.
- Incorrect superblock summary counts.

Requirements for Checking

The **fsck** program will refuse to check any file system for which any of the following conditions hold true:

- The file system is mounted
- The special file associated with the file system cannot be opened.
- The specified pathname (or its device node associate in **/etc/fstab**) is not a block-special, character-special, or regular file whose size can be determined.

Fsck Output

If the **-p** option is used, **fsck** prints out one or more lines for each inconsistency it corrects, indicating the file system fixed and the error corrected. After successfully checking or correcting a file system, **fsck** prints out the name of the file system, the number of files on it, and the number of free and used blocks.

If the **-p** option is not used, **fsck** is more verbose. It will first print out the name of the file system. Then **fsck** prints a message as it enters each phase of checking a file system. A message is printed for each inconsistency encountered, and the operator is prompted for approval before each correction is attempted. (If the **-y** or **-n** flags are used, **fsck** automatically answers such prompts itself.) When checking is complete for the file system, a message is printed if any corrections were made. Finally, the numbers (used, free and total) of files and blocks are printed.

The **fsck** program attempts to give as much information as possible about any files for which you must make decisions (such as whether to remove it, etc.). At least the following information will always be displayed:

- I-number
- 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 groupname

Fsck Error Conditions

When **fsck** detects an inconsistency, it reports the error condition to the operator. If a response is required, **fsck** prints a prompt message and waits for a response. This appendix explains the meaning of each error condition, possible responses, and related error conditions.

The error conditions are organized by the phase of the **fsck** program in which they can occur. The error conditions that may occur in more than one phase are discussed under "General."

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. Several different abbreviations are also used:

B	A (decimal) disk block number.
N	A decimal number.
O	An octal number.
C	A character.
D, F	A directory name, filename 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.

General

The messages described in this section may appear at any time during an **fsck** session.

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. **Fsck** terminates.

Cannot read block B

A disk read of block number B has failed. **Fsck** 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. **Fsck** continues execution, but the file system being checked is not marked as mountable upon conclusing of checking. Use **diskman** to remap the bad block B and run **fsck** again.

Cannot allocate memory for internal tables (N bytes requested)

Fsck cannot allocate enough memory; this can only occur during stand-alone **fsck**. **Fsck** will abort. Bring up your system and use the **fsck** command instead.

Invalid response; please answer yes or no

An invalid answer has been entered in response to one of **fsck**'s questions. **Fsck** 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.

Fork failed

Fsck 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.

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 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. **Fsck** will abort.

Unknown option: -C

An unknown option flag, C, has been specified on the **fsck** command line. The valid flags are: **-y**, **-n**, **-p**, **-q**, **-t**, **-D**, **-f**, **-s**, **-S** and **-x**. **Fsck** will abort.

The directory D is the mount point for F

Fsck 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.

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.

F is not a regular file, block-special file, character-special file or valid mount point

Fsck 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. **Fsck** will abort checking this file system.

Cannot open F for reading

Fsck has been given a file system F to check, but F cannot be opened for reading. Check the mode of F. **Fsck** will abort checking this file system.

Cannot open F for writing

Fsck has been given a file sytem 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. **Fsck** will abort checking this file system.

Cannot determine disk size of F

Fsck has been given a file sytem 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. **Fsck** will abort checking this file system.

Cannot read superblock copy N

One of the two superblock copies cannot be read. **Fsck** will attempt to use the other copy and continue.

Cannot find a readable copy of the superblock

Neither of the two copies of the superblock can be read. **Fsck** will abort checking this file system.

Superblock copy N is invalid

One of the two superblock copies does not contain the required self-identification information. **Fsck** will attempt to use the other copy and continue.

Cannot find a valid copy of the superblock

Neither of the two copies of the superblock contain the required self-identification information. **Fsck** will abort checking 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. **Fsck** will use the first copy (which is guaranteed to be more recent) and continue.

Superblock has invalid contents in reserved area -- fix?

A copy of the superblock has non-zero contents in a reserved area. If running with the **-p** flag, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES	The superblock's reserved area is zeroed out.
NO	Ignore this error condition. **Fsck** will not mark this file system as mountable upon completing the check.

No check necessary for F

The file system F is already marked mountable and **fsck** was invoked with the **-x** flag. **Fsck** will not check this file system.

Invalid Disk Allocation Region size: N blocks

The DAR size stored in the superblocks is invalid. **Fsck** 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. **Fsck** will abort checking this file system.

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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** will abort checking this file system.

Invalid second allocation threshold file size: N -- fix?

The superblocks contain an invalid second allocation threshold file size (the number of blocks a file can allocate in a non-initial DAR before all subsequent allocations are made from a different DAR). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the second allocation threshold file size to the default limit for DARs of the size specified in the superblock.
NO	Ignore this error condition. **Fsck** will abort checking this file system.

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. **Fsck** will not mark this file system as mountable upon completing the check.

File system is too large to check

Stand-alone **fsck** cannot allocate enough memory for its internal tables to begin checking the file system. **Fsck** will abort checking this file system. Bring up your system and use the **fsck** command instead.

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. **Fsck** will not mark this file system as mountable upon completing the check.

Phase 1 - Check Blocks and File Sizes

This phase 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.

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 zeroing inode I's reserved area.

NO Ignore this error condition. **Fsck** 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. **Fsck** 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. **Fsck** will abort checking this file system.

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. **Fsck** will not mark this file system as mountable upon completing the check.

Inode I claims an invalid block (B) -- clear bad pointer?

The inode I claims block B, which does not exist. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES	Fix this error condition by clearing the pointer in inode I that claims the non-existent block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.
NO	Ignore this error condition. **Fsck** 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

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. **Fsck** will not mark this file system as mountable upon completing the check.

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.

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.

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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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 filename 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with an illegal filename: F -- remove bad directory entry?

The directory inode I1 has a directory entry for inode I2, but the entry's name F is "." or "..". These two names are reserved for the directory's links to itself and to its parent, respectively. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode I2. If inode I2 is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. **Fsck** 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has invalid contents in its reserved area -- fix?

The directory inode I1 has a directory entry for inode I2, which has non-zero information in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the contents of the reserved area of inode I2.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by zeroing out the reserved area of inode I2.

NO Ignore this error condition. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

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.

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.

Could not reconnect inode I

Fsck 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.

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.

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. **Fsck** will not mark this file system as mountable upon completing the check.

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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

Directory inode I is already as large as it can become

Fsck has discovered that a directory it was attempting to expand is already the maximum size a directory can become.

Cannot find enough contiguous free blocks to expand directory inode I

Fsck 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.

Inode I1 lists as its space parent inode number I2, which is not a valid control point directory -- reset space parent to root?

The inode I1 has the non-control point directory inode I2 listed as its space parent. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset I1's space parent to inode 2, the root of the file system.

Possible responses to the `reset?` prompt are:

YES	Fix this error condition by resetting I1's space parent to inode 2, the root of the file system.
NO	Ignore this error condition. **Fsck** will not mark this file system as mountable upon completing the check.

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.

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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

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. **Fsck** will not mark this file system as mountable upon completing the check.

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. **Fsck** will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has invalid contents in its reserved area -- fix?

Disk allocation region number N has non-zero contents in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to zero out the reserved area.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by zeroing the contents of the reserved area.
NO	Ignore this error condition. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** 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. **Fsck** will not mark this file system as mountable upon completing the check.

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

Fsck has successfully completed checking the file system and it has been marked as mountable.

File System is still inconsistent and not mountable

Fsck 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.

Unconnected files still remain. Mount the file system and remove files to free data blocks and inodes

Fsck 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 can be recovered by running **fsck** again after enough blocks and inodes have been freed to allow them room to be reconnected.

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.

End of Appendix

Appendix E
Expert UUCP Information

This appendix gives further information on some of the topics in Chapter 12, UUCP Management. Those topics are:

- UUCP Connections
- UUCP Data Files
- UUCP Cleanup

UUCP Connections

Before your computer can communicate with a remote computer, you must set up a two-way communication connection between the machines. This section describes the two kinds of UUCP connections.

Direct Connection

This method requires a direct connection from a port on a local computer to a port on the remote computer. A direct line is advantageous when communication is required with the remote computer on a regular basis. The link is always available and access time is short. The disadvantage of the direct link is that the port cannot be used for anything else. The 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 or less. Longer lengths can be obtained by using a lower transmission rate and/or limited distance modems at both ends of the link.

Direct connections are beneficial only when:

- It is not possible to link the computers together through a Local Area Network (LAN).
- Two computers transfer large amounts of data on a regular basis.
- Two computers are located no more than several hundred cable feet apart.

The distance between two directly linked computers is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet 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 cable feet to connect the two computers or communications will be unreliable. This link should operate comfortably at 9600 bits per second in a clean (noise free) environment.

If the two computers are separated by more than 100 feet of cable, you must use a null-modem cable.

Dial-up Connection

In this case, the computer that is going to make the connection would call the remote computer using an Automatic Calling Unit (ACU). The remote computer answers via its own ACU and makes the connection. With this arrangement, the ports are not dedicated to only one computer. A dial-up link also requires more hardware (such as the ACU) than the direct connection. Transmission rates are limited to the capacity of the ACUs.

Another important fact to consider when setting up devices is the type of controller (such as an IAC) you intend to use. In general, you should connect direct lines to IAC16 boards and modem lines to IAC8 boards. If your system does not use IAC boards, check your hardware documentation to see which ports are for modems and which are for terminal connections.

Refer to your modem documentation for information on configuring your modem for dial-out or dial-in use. In **/usr/lib/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 machine's language and command syntax.

Modem and Direct Link Support Files

If you make changes manually, be sure to update the following support files to reflect the presence of a direct link or a modem connection:

- **/usr/lib/uucp/Devices**
- **/etc/inittab**
- **/usr/lib/uucp/Systems**.

Additionally, the **/usr/lib/uucp/Dialers** file must contain information on any modem you use.

When you have determined which communication links best suit your needs, you will need to dedicate one tty line to each communication link you wish to use, unless you run **uugetty** on a line. In this case, the line may be used for both dialing in and dialing out.

UUCP Data Files

UUCP data files must be owned by **nuucp** and must have read and write access permissions. The following sections describe the files in **/usr/lib/uucp** that support UUCP file transfers. The files discussed are:

Devices	Dialers
Systems	Dialcodes
Permissions	Poll
Sysfiles	Maxuuxqts
Maxuuscheds	remote.unknown

Devices File

The **Devices** file (**/usr/lib/uucp/Devices**) contains information for all the devices that may be used to establish a link to a remote computer; these are devices such as automatic call units, direct links, and network connections.

NOTE: This file works closely with the **Dialers, Systems,** and **Dialcodes** files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the **Devices** file has the following format:

Type Line Line2 Class Dialer-Token-Pairs

Each of these fields is defined in the following section.

Type This field may contain one of two keywords (**Direct** or **ACU**), the name of a Local Area Network switch, or a system name.

Direct This keyword indicates a Direct Link to another computer or a switch (for **cu** connections only).

ACU This keyword indicates that the link to a remote computer is made through an automatic call unit (also known as an automatic dial modem). This modem may be connected either directly to your computer or indirectly through a Local Area Network (LAN) switch.

LAN_Switch

This value can be replaced by the name of a LAN switch. Micom and Develcon are the only ones for which there are caller scripts in the **Dialers** file. You can add your own LAN switch entries to the **Dialers** file.

Sys-Name

This value indicates a direct link to a particular computer. (*Sys-Name* is replaced by the name of the computer.) This naming scheme is used to convey the fact that the line associated with this **Devices** entry is for a particular computer in the **Systems** file.

The keyword used in the *Type* field is matched against the third field of **Systems** file entries as shown below:

```
Devices: ACU tty11 - 1200 penril

Systems:  eagle Any ACU  1200 3251 ogin: uucp \
          ssword: Oakgrass
```

Line This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the ACU for a particular entry was attached to the **/dev/tty11** line, the name entered in this field would be **tty11**.

Line2 If the keyword **ACU** appears in the *Type* field and the ACU is an 801 type dialer, *Line2* would contain the device name of the 801 dialer. (801 type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line, defined in the *Line* field.) This means that one line would be allocated to the modem and another to the dialer. Since non-801 dialers will not normally use this configuration, the *Line2* field will be ignored by them, but it must still contain a hyphen (-) as a placeholder.

Class If the keyword **ACU** or **Direct** is used in the *Type* field, *Class* may be just the speed of the device. However, it may contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). This is necessary because many larger offices may have more than one type of telephone network: one network may be dedicated to serving only internal office communications while another handles the external communications. In such a case, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The keyword used in the *Class* field of the **Devices** file is matched against the fourth field of **Systems** file entries as shown below:

```
Devices: ACU tty11 - D1200 penril

Systems: eagle Any ACU D1200 3251 ogin: nuucp \
              ssword: Oakgrass
```

Some devices can be used at any speed, so the keyword **Any** may be used in the *Class* field. If **Any** is used, the line will match any speed requested in a **Systems** file entry. If this field is **Any** and the **Systems** file *Class* field is **Any**, the speed defaults to 1200 bps.

Dialer-Token-Pairs:

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, or it may be **direct** for a Direct Link device. You can have any number of Dialer-Token-Pairs. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the **Systems** file.

This field has the format:

dialer token dialer token

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair contains only a *dialer* portion and 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 sequence describes the token *but is not the token*. See below.

1. If an automatic dialing modem is connected directly to a port on your computer, the *DTP* field of the associated **Devices** file entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** file entry with an entry in the **Dialers** file. Therefore, the *dialer* field must match the first field of a **Dialers** file entry as shown below:

   Devices: `ACU tty11 - 1200` **ventel**

   Dialers: ventel `=&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!`

   Notice that only the *dialer* portion (**ventel**) is present in the *DTP* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry.

2. If a direct link is established to a particular computer, the *DTP* field of the associated entry would contain the keyword **direct**. This is true for both types of direct link entries, **Direct** and *System-Name* (refer to discussion on the *Type* field).

3. If a computer with which you wish to communicate is on the same local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry as shown below:

   Devices: `develcon tty13 - 1200` **develcon** `\D`

   Dialers: **develcon** `"" "" \pr\ps\c est:\007 \E\D\e \007`

   As shown, the *token* portion is left blank, which indicates that it is retrieved from the **Systems** file. The **Systems** file entry for this particular computer will contain the token in the *Phone* field, which is normally reserved for the phone number of the computer (refer to **Systems** file, *Phone* field). This type of *DTP* contains an escape character (**\D**), which ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the **Dialcodes** file.

4. If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch will make the connection to

the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the **Dialers** file as shown below:

```
Devices:  ACU tty14 - 1200 develcon vent ventel

Dialers:  develcon "" "" \pr\ps\c est:\007 \E\D\e \007
Dialers:  ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

In the first pair, **develcon** is the dialer and **vent** is the token that is passed to the Develcon switch to tell it which device (ventel modem) to connect to your computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the ventel modem has been connected, the second pair is accessed, where ventel is the dialer and the token is retrieved from the **Systems** file.

There are two escape characters that may appear in a *DTP* field:

\T Indicates that the *Phone* (*token*) field should be translated using the **Dialcodes** file. This escape character is normally placed in the **Dialers** file for each caller script associated with an automatic dial modem (penril, ventel, etc.). 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 (**/usr/lib/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 is transmitted and expected (called a chat script), and it is often used to dial a phone number using an ASCII dialer (such as the automatic dial modem).

As shown earlier, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801). Here an attempt is made to match the fifth field in the **Devices** file with the first field of each **Dialers** file entry. In addition, each odd numbered **Devices** field (the token field) starting with the seventh position is used as an index into the **Dialers** file. If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations. Each entry in the **Dialers** file has the following format:

dialer substitutions expect-send ...

The *dialer* field matches the fifth and additional odd numbered fields in the **Devices** file. The *substitutions* field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate = and - into whatever 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 :--: ato12=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
```

There are also three AT&T modems that 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. The meanings of some of the escape characters (those beginning with "\") used in the **Dialers** file are listed below:

\p	pause (approximately ¼ to ½ second)
\d	delay (approximately 2 seconds)
\D	phone number or token without **Dialcodes** translation
\T	phone number or token with **Dialcodes** translation
\K	insert a BREAK

\E	enable echo checking (for slow devices)
\e	disable echo checking
\r	carriage return
\c	no New Line or carriage return
\n	send New Line
\nnn	send octal number.

Additional escape characters that may be used are listed in the section discussing the **Systems** file.

The Penril entry in the **Dialers** file is executed as follows. First, the phone number argument is translated, replacing any **=** with a **W** (wait for dialtone) and replacing any **-** with a **P** (pause). The handshake given by the remainder of the line works as follows:

"	Wait for nothing. (In other words, proceed to the next thing.)
\d	Delay for 2 seconds.
>	Wait for a **>**.
K\c	Send a **K**. Send no terminating New Line
:	Wait for a **:**.
\EP\T	Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send a **P** and the phone number. The **\T** means take the phone number passed as an argument and apply the **Dialcodes** translation and the modem function translation specified by field 2 of this entry.
OK	Waiting for the string **OK**.

Systems File

The **Systems** file (**/usr/lib/uucp/Systems**) contains the information needed by the **uucico** daemon to establish a communication link to a remote computer. Each entry in the file represents a computer that can be called by your computer. In addition, UUCP software can be configured to prevent any computer that does not appear in this file from logging in on your computer. More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequence.

Using the **Sysfiles** file, you can define several files to be used as "Systems" files. See the description of the **Sysfiles** file later in this appendix for details. Each entry in the **Systems** file has the following format:

System-name Time Type Class Phone Login

Each of these fields is defined in the following section.

System-name

This field contains the host name of the remote computer.

Time

This field is a string that indicates the day-of-week and time-of-day when the remote computer can be called. The format of the *Time* field is:

daytime[*;retry*]

The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa	for individual days
Wk	for any week-day (Mo Tu We Th Fr)
Any	for any day
Never	for a passive arrangement with the remote computer. If the *Time* field is **Never**, your computer will never initiate a call to the remote computer. The call must be initiated by the remote computer. In other words, your computer is in a passive mode with respect to the remote computer (see discussion of **Permissions** file).

Here is an example:

```
Wk 1700-0800, Sa, Su
```

This example allows calls from 5:00 p.m. to 8:00 a.m., Monday through Friday, and calls any time Saturday and

Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The *time* portion should be a range of times such as 0800-1230. If no *time* portion is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, **0800-0600** means all times are allowed other than times between 6 a.m. and 8 a.m. An optional subfield, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, **Any;9** is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs.

Type This field contains the device type that should be used to establish the communication link to the remote computer. The keyword used in this field is matched against the first field of **Devices** file entries as shown below:

```
Systems:  eagle Any ACU,g D1200 3251 ogin: nuucp \
            ssword: Oakgrass

Devices:  ACU tty11 - D1200 penril
```

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol **g** to the device type **ACU**. See the information under the "Protocols" section in the description of the **Devices** file for details.

Class This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *Class* field). Some devices can be used at any speed, so the keyword **Any** may be used. This field must match the *Class* field in the associated **Devices** file entry as shown below:

```
Systems:  eagle Any ACU D1200 NY3251 ogin: nuucp \
            ssword: Oakgrass

Devices:  ACU tty11 - D1200 penril
```

If information is not required for this field, use a **-** as a place holder for the field.

Phone This field is used to provide the phone number (token) of the remote computer for automatic dialers or LAN switches. The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

```
Systems:  eagle Any ACU D1200 NY3251 ogin: nuucp \
            ssword: Oakgrass

Dialcodes:  NY 9=1212555
```

In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 4 seconds before dialing the next digit.

If your computer is connected to a LAN switch, you may access other computers that are connected to that switch. The **Systems** file entries for these computers will not have a phone number in the *Phone* field. Instead, this field will contain the token that must be passed on to the switch so it will know which computer your computer wishes to communicate with. (This is usually just the system name.) The associated **Devices** file entry should have a **\D** at the end of the entry to ensure that this field is not translated using the **Dialcodes** file.

Login This field contains login information given as a series of fields and subfields of the format:

expect send

where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received.

The *expect* field may be made up of subfields of the form:

expect[*-send-expect*]...

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string. For example, with **login--login**, UUCP will expect **login**. If UUCP gets **login,** it will go on to the next field. If it does not get **login**, it will send a null string followed by a New Line, then look for **login** again. If no characters are initially expected from the remote computer, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a New Line unless the *send* string is terminated with a **\c**.

Here is an example of a **Systems** file entry that uses an expect-send string:

```
owl Any ACU 1200 Chicago6013 "" \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**, then when you get **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. The following escape characters are useful in UUCP communications:

\N	Send or expect a null character (ASCII NUL).
\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.
\p	Pause for approximately ¼ to ½ second.
\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.)
\e	Echo check off.
\n	Send a new-line character.
\r	Send or expect a carriage-return.
\s	Send or expect a space character.
\t	Send or expect a tab character.
\\	Send or expect a \ character.
EOT	Send or expect EOT new-line twice.
BREAK	Send or expect a break character.
\K	Same as BREAK.
\ddd	Collapse the octal digits (ddd) into a single character.

Dialcodes File

The **Dialcodes** file (**/usr/lib/uucp/Dialcodes**) contains the dial-code 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 (**/usr/lib/uucp/Permissions**) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Another option is available that specifies the commands that a remote site can execute on the local computer. Note that the **Permissions** prototype file sent with this software release is most restrictive.

Permissions File Entries

Each entry is a logical line with physical lines terminated by a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair in the following format:

name=value

Note that no white space is allowed within an option assignment.

Comment lines begin with a "#" and they occupy the entire line up to a New Line character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** file entries:

LOGNAME	Specifies the permissions that take effect when a remote computer logs in on (calls) your computer.
MACHINE	Specifies permissions that take effect when your computer logs in on (calls) a remote computer.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option.

Considerations

The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote computers:

- Each login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry will have the following default permissions/restrictions:
    1) Local send and receive requests will be executed.
    2) The remote computer can send files to your computer's **/var/spool/uucppublic** directory.
    3) The commands sent by the remote computer for execution on your computer must be one of the default commands; usually **rmail**.

Options

This section describes each option, specifies how they are used, and lists their default values.

REQUEST

When a remote computer calls your computer and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote computer can request to set up file transfers from your computer. The string

```
REQUEST=yes
```

specifies that the remote computer can request to transfer files from your computer. The string

```
REQUEST=no
```

specifies that the remote computer cannot request to receive files from your computer. This is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry. A note on security: When a remote machine calls you, unless you have a unique login and password for that machine you don't know if the machine is who it says it is.

SENDFILES

When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it. The SENDFILES option specifies whether your computer can send the work queued for the remote computer.

The string

```
SENDFILES=yes
```

specifies that your computer may send the work that is queued for the remote computer as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if your computer is in a "passive mode" with respect to the remote computer.

The string

```
SENDFILES=call
```

specifies that files queued in your computer will be sent only when your computer calls the remote computer. The call value is the default for the SENDFILES option. This option is only significant in LOGNAME entries since MACHINE entries apply when calls are made out to remote computers. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that **uucico** can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the **uucppublic** directory as shown in the following strings:

```
READ=/var/spool/uucppublic
WRITE=/var/spool/uucppublic
```

The strings

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a colon separated list of path names. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out.

To grant permission to deposit files in **/usr/news** as well as the public directory, the following values would be used with the WRITE option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

It should be pointed out that if the READ and WRITE options are used, all path names must be specified because the path names are not added to the default list. For instance, if the **/usr/news** path name was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote computers to be able to write over your **/etc/passwd** file so **/etc** shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

```
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic
```

would permit reading any file except those in the **/etc** directory (and its subdirectories—remember, these are prefixes) and writing only to the default **/var/spool/uucppublic** directory. NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. There are two examples of when you would use CALLBACK. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that will be billed for the longer call.

The string

```
CALLBACK=yes
```

specifies that your computer must call the remote computer back before any file transfers will take place.

The default for the CALLBACK option is

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a conversation will never get started.

COMMANDS

The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote computer. Files and a command are sent to the target computer for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote computer can execute on your computer. Note that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries define command permissions whether we call the remote system or it calls us.

The string

```
COMMANDS=rmail
```

indicates the default commands that a remote computer can execute on your computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:mail:lp
```

overrides the COMMAND default so that the computers owl, raven, hawk, and dove can now execute **rmail**, **mail**, and **lp** on your computer.

In addition to the names as specified above, there can be full path names of commands. For example,

```
COMMANDS=rmail:/usr/local/mail:/usr/bin/lp
```

specifies that command **rmail** uses the default path. The default paths for your computer are **/bin**, **/usr/bin**, and **/usr/local**. When the remote computer specifies **mail** or **/usr/bin/mail** for the command to be executed, **/usr/local/mail** will be executed regardless of the default path. Likewise, **/usr/bin/lp** is the **lp** command that will be executed.

Including the ALL value in the list means that any command from the remote computer(s) specified in the entry will be executed. If you use this value, you give the remote computer full access to your computer. BE CAREFUL. This allows far more access than normal users have.

The string

```
    COMMANDS=/usr/local/mail:ALL:/usr/bin/lp
```

illustrates two points: The ALL value can appear anywhere in the string, and the path names specified for **mail** and **lp** will be used (instead of the default) if the requested command does not contain the full path names for **mail** or **lp**.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like **cat** and **uucp** are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE

The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged computers have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure. (VALIDATE is merely an added level of security on top of the COMMANDS option, though it is a more secure way to open command access than ALL.)

Careful consideration should be given to providing a remote computer with a privileged login and password for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer.

The LOGNAME entry

```
    LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote computers that claims to be eagle, owl, or hawk logs in on your computer, it must have used the login **uucpfriend**. As can be seen, if an outsider gets the **uucpfriend** login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? It links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote computer is logged in. In fact, it is an asynchronous process with no knowledge of what computer sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote computer has its own "spool" directory on your computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote computer are put in its spool directory after being transferred to your computer. When the **uuxqt** daemon runs, it can use the spool directory name to find the MACHINE entry in the **Permissions** file and get the COMMANDS list, or if the computer name does not appear in the **Permissions** file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/mail \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The value in the COMMANDS option means that remote mail and **/usr/local/mail** can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the computers listed, you are really calling either **eagle**, **owl**, or **hawk**. Therefore, any files put into one of the **eagle**, **owl**, or **hawk** spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these three computers, its execution files will also be put in the privileged spool directory. You therefore have to validate that the computer has the privileged login **uucpz**.

You may want to specify different option values for the computers your computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many computers calling in, and the command set changes from time to time. The name "OTHER" for the computer name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:mail:/usr/local/Photo:/usr/local/xp
```

All other options available for the MACHINE entry may also be set for the computers that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Poll File

The **Poll** file (**/usr/lib/uucp/Poll**) contains information for polling remote computers. Each entry in the **Poll** file contains the name of a remote computer to call, followed by a tab character (a space won't work), and the hours the computer should be called. The format of entries in the **Poll** file are:

sys-name hour ...

For example the entry:

```
eagle    0 4 8 12 16 20
```

will provide polling of computer **eagle** every four hours.

The **uudemon.poll** script does not actually perform the poll. It merely sets up a polling work file (always named C.*file*), in the spool directory that will be seen by the scheduler, which is started by **uudemon.hour**.

Sysfiles File

The **/usr/lib/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; and *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. A backslash-carriage return (**\<CR>**) 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 **/usr/lib/uucp/Sysfiles**, then both **uucico** and **cu** will first look in **/usr/lib/uucp/Systems**. If the system they're trying to call doesn't have an entry in that file, or if the entries in the file fail, then they'll look in **/usr/lib/uucp/Local_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

This 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

This 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

This file is a shell script that executes when a machine that is not in the **Systems** file attempts to start a conversation. It will log the conversation attempt into the file **/var/spool/uucp/.Admin/foreign** and fail to make a connection. If you change the permissions of this file so it cannot execute (**chmod 000 remote.unknown**), your system will accept any conversation requests.

UUCP Spool Files

The files described in this section are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM Temporary date file. These data files are created by UUCP processes under the spool directory (i.e., **/var/spool/uucp/***X*) when a file is received from another computer. The directory *X* has the same name as the remote computer that is sending the file. The names of the temporary data files have the format:

TM.*pid.ddd*

where *pid* is a process-ID and *ddd* is a sequential three digit number starting at 0.

When the entire file is received, the **TM.***pid.ddd* file is moved to the path name specified in the **C.***sysnxxxx* file (discussed below) that caused the transmission. If processing is abnormally terminated, the **TM.***pid.ddd* file may remain in the *X* directory. These files will be automatically removed by **uucleanup**.

LCK Lock file. Lock files are created in the **/var/spool/locks** directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The names of lock files have the format:

LCK..*str*

where *str* is either a device or computer name. These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

C.*name* Work file. Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote computer.

The names of work files have the format:

C.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work; the **uucico** code sets this priority and you may change it with **uucp(1)** and **uux(1)**. *xxxx* is the four digit job sequence number assigned by **uucp**. Work files contain the following information:

- Full 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 computer, *xxxx* is a four-digit job sequence number assigned by **uucp**. The four digit job sequence number may be followed by a sub-sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X.*name* Execute file. Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

X.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is a four digit sequence number assigned by **uucp**. Execute files contain the following information:

- Requester's login and computer name.
- Name of file(s) required for execution.
- Input to be used as the standard input to the command string.
- Computer and file name to receive standard output from the command execution.
- Command string.
- Option lines for return status requests.

Log Files

Log files are created for each remote machine with which your computer communicates. There are directories for each of the **uucico, uucp, uux** and **uuxqt** commands with subdirectories under these for each machine making requests. The logfiles are kept in the directory **/var/spool/uucp/.Log**. These logfiles are combined and stored in the directory **/var/spool/uucp/.Old** when **uudemon.cleanup** is executed. The combined files are kept three days before they are removed. If space is a problem, the administrator may consider reducing the number of days the files are kept.

The information from the individual log files for each machine and each program (e.g., machine *dumbo* has a logfile for **uucico** requests and a logfile for **uuxqt** execution requests) can be accessed with the **uulog** program. These files are combined and stored in directory **/usr/lib/uucp/.Old** whenever **uudemon.cleanup** is executed. This shell script saves files that are two days old. The two days can be easily modified in the **uudemon.cleanup** shell. If space is a problem, you might consider reducing the number of days the files are kept.

UUCP File Cleanup

The **uustat** program should be invoked regularly to provide information about the status of connections to various machines and the size and age of the queued requests. The **uudemon.admin** shell should be started by **cron** at least once per day to send the administrator the current status. Of particular interest are the the age (in days) of the oldest request in each queue, the number of time a failure has occurred when attemptingto reach that machine, and the reason for the failure. In addition, the age of the oldest execution request (**X.file**) is also given.

Execution files older that a few days can probably be deleted since the only reason they have not been executed is because data files required for execution were not sent. These files are usually sent at the same time as the **X.file**, so the problem is likely at the other end.

The **uucleanup** program, which is run from **uudemon.cleanup** removes these files. Options to **uucleanup** specify the age for sending a warning message to the requester and age for deleting various files. Before deleting, the program tries to figure out what the job was and, if possible, tries to send it to the receiver. If this is not possible, it is returned to the sender.

Public Area Cleanup

To keep the local file system from overflowing when files are sent to the public area, the **uudemon.cleanup** procedure is set up with a **find** command to remove any files that are older than seven days and directories that are empty. The interval may need to be shortened if there is not sufficient space to devote to the public area.

End of Appendix

Appendix F Glossary

DG/UX Terms

Other parts of this manual have definition sections. If you don't see a term defined here, check the Index for the term.

address	A number, label, or name that indicates the location of information in the computer's memory.
alias	A mailing list containing one or more login names. Mail addressed to an alias goes to all members on the alias list. Aliases are listed in **/usr/lib/aliases.**
a.out	The default name of a compiled object file; historically a.out signifies assembler output.
archive	A collection of data gathered from several files into one file (doing backups); especially, such a collection gathered by **ar**(1) for use as a library.
ASCII	An acronym for American Standard Code for Information Interchange, a standard for data representation that is followed in the DG/UX system. ASCII code represents alphanumeric characters as binary numbers. The code includes 128 upper- and lower-case letters, numerals, and special characters. Each alphanumeric and special character has an ASCII code (binary) equivalent that is 7 bytes long.
automatic calling unit	ACU. This is a hardware device used to dial stored telephone numbers; it allows the system to contact another system over phone lines without manual intervention.
bad block	A block or section of a storage medium that cannot store data reliably.
baud rate	The transmission speed in bits per second at which information passes between a terminal and a computer.

block	The basic unit of buffering in the kernel, 512 bytes.
block device	A device upon which a file system can be mounted, typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by blocks.
boot	To start the operating system, so called because the kernel must bootstrap itself from secondary storage into an empty machine.
boot program	A program that loads the operating system into main memory.
buffer	A space in computer memory that saves information temporarily. An editor buffer is an area where a text editor keeps a copy of the file being edited. When editing ends, the contents of the buffer are erased.
buffer pool	A region of storage available to the file system for holding blocks; all but raw input-output for block devices goes through the buffer pool so read and write operations may be independent of device blocks.
byte	A unit of storage in the computer. On DG/UX systems, a byte is eight bits (binary digits), the equivalent of one character of text.
cartridge tape	This is a storage medium that consists of a magnetic tape wound on spools housed in a plastic container.
central processing unit	CPU. The hardware that controls everything that the computer does. It interprets the machine language instructions created by the compiler or interpreter. These machine language instructions are 0's and 1's that are transmitted to the CPU as electric pulses. The electric pulses affect the chips and transistors of the CPU. Most times, the first programs executed by a user will be the first executed by the CPU. The first command waiting to be executed has the highest priority.
character device	A device upon which a file system cannot be mounted such as a terminal or the null device.
chat script	A sequence of commands that determines how UUCP communication is established between machines. See the chat scripts in **/usr/lib/uucp/Dialers**.

command	An instruction to the shell, usually to run a program as a child process.
compiler	A program that transforms high-level language instructions (source code) into object code or assembly language. Assembly language code may then be passed to the assembler for further translation into machine instructions. The C compiler converts programs into assembly language programs that are eventually translated into object files by the assembler.
configuration	The arrangement of the software and hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.
controller	A device that directs the transmission of data over the data links of a network.
core file	A core image of a terminated process saved for debugging; a core file is created under the name 'core' in the current directory of the process.
core image	A copy of all the segments of a running or terminated program; the copy may exist in main store, in the swap area, or in a core file. See **signal(2)**.
crash	To terminate abruptly. If a hardware or software error condition develops that the system can't handle, it takes itself out of service, or crashes. Such conditions occur when the system can't allocate resources, manage processes, respond to requests for system functions, or when the electrical power is unstable.
cron	A program that creates daemons to invoke commands at specified dates and times. See **cron(1M)**.
cylinder	The set of all tracks on a disk that are the same distance from the axis about which the disk rotates.
daemon	A background process, often perpetual, that performs a system-wide public function, e.g. **calendar(1)** and **cron(1M)**.
device	A file that is not a plain file or a directory, such as a tape drive, or the null device; a special file. Device files often represent a physical input/output unit.

device driver	A set of I/O subroutines in a table within the kernel. Device drivers are either character-oriented or block-oriented so that a program can access a device in either character or block mode. Programs invoke device drivers. For each device, a device driver specifies open, close, read, write, etc.
diagnostic	A message printed at your terminal that identifies and isolates program errors.
directory	A file that comprises a catalog of filenames; the organizing principle of the file system. A directory consists of entries which specify further files (including directories), and constitutes a node of the directory tree.
directory entry	An association of a name with an inode number appearing as an element of a directory.
directory hierarchy	The tree of all directories, in which each is reachable from the root via a chain of subdirectories.
disk	A platter coated with magnetic material on which data can be electronically stored. Usually referred to as a physical disk.
diskette	A magnetic storage medium which is smaller and more flexible than a hard disk. Also called a floppy disk.
domain	A logical grouping of hosts in a Network File System environment. Each host in a domain relies on the same domain name server(s) for certain resource sharing and security services. Each domain has one primary and zero or more secondary domain name servers.
domain name server	A computer that creates and maintains the following information for hosts in a Network File System domain: advertised resources, host names and passwords, names and addresses for name servers of other domains (optional), host user and group information used for ID mapping (optional).
drive	The hardware device that holds magnetic disks, diskettes, and tapes while they are in use.
dump	To write system memory and/or the system kernel image to tape.

dump cycle	A scheme for doing daily, weekly, and monthly backups on tape.
environment	A set of strings, distinct from the arguments, made available to a process when it executes a file. The environment is usually inherited across **exec(2)** operations. A specific environment is maintained by the shell. The shell environment is composed of variables whose values define the way a user interacts with the system.
error	Occurs when a hardware or software condition prevents the successful execution of a system or a user process.
executable file	An object file that is ready to be copied into the address space of a process to run as the code of that process; a file that has execute permission, either an executable file or a shell script.
FIFO	A FIFO is a named permanent pipe which allows two unrelated processes to exchange information using a pipe connection. FIFO stands for first in-first out.
file	In general, a file is a potential source of input or destination for output. The DG/UX system sees a file as an inode that specifies whether the file is a plain file, a special file, or a directory.
file descriptor	A conventional integer quantity that designates an open file. The number is assigned by the operating system when a process opens a file.
file system	A collection of files that can be mounted on a directory. Each file of a file system appears exactly once in the inode list of the file system and is accessible via some path from the root directory of the file system.
filter	A program that reads from the standard input and writes on the standard output, so called because it can be used as a data-transformer in a pipeline.
flag	An indicator used on a command line to signal a specific condition to a command or to request particular processing. DG/UX system flags are usually indicated by a leading hyphen. The word *option* is sometimes used interchangeably with flag.

flush	To empty a buffer, for example to throw away unwanted input-output upon interrupt.
fork	To split one process into two, the parent process and child process, with separate, but initially identical, text, data, and stack segments.
formatting	The process of imposing an addressing scheme on a disk. This includes the establishment of a Primary System Area and the mapping of the disk into tracks and sectors.
free list	In a file system, a bitmap indicating which blocks are not occupied by data.
getty	A program that is one of a series of processes which connect the user to the DG/UX system. **getty** is invoked by **init(1M)**, and in turn invokes **login(1)**. See **getty(1M)**.
group ID	An integer value, usually associated with one or more login names; as the user ID of a process becomes the owner of files created by the process, so the group ID of a process becomes the group of such files. See Chapter 11.
host	A computer that is configured to share resources with other computers in a network.
syac	Intelligent asynchronous controller. A terminal controller board containing 8 or 16 terminal lines. See **iac(7)**.
ILC	Intelligent LAN Controller. An interface to the Ethernet network that allows for dynamic mapping between Internet and Ethernet addresses on a network. See **ilc(6)**.
init	A general process spawner which is invoked as the last step in the boot procedure; it regularly checks a table that defines what processes should run at what run level. See **init(1M)**.
inode	An element of a file system; an inode specifies all properties of a particular file and locates the file's contents, if any.
integrity	In a file system, the quality of being without errors due to bad blocks.

interface programs	Shell scripts furnished with the LP system software which interface between the user and the printer.
interrupt	A break in the normal flow of a system or program. Interrupts are initiated by signals that are generated by a hardware condition or a peripheral device indicating that a certain event has happened. When the interrupt is recognized by the hardware, an interrupt handling routine is executed. An interrupt character is a character (normally ASCII) that, when typed at a terminal, causes an interrupt.
kernel	The nucleus of the DG/UX system. It controls access to the computer, manages the computer's memory, maintains the file system, and allocates the computer's resources among users. The kernel is sometimes described as the DG/UX system proper; resident code that implements the system calls.
kernel address space	A portion of memory used for data and code addressable only by the kernel.
line discipline	A module to handle protocol or data conversion for a stream. A line discipline, unlike a filter, is part of the kernel.
link count	The number of directory entries that pertain to an inode; a file ceases to exist when its link count becomes zero and it is not open.
load device	Designates the physical device from which a program will be loaded into main memory.
log file	A file containing records of transactions that occur on the system; software that spools, for example, generates various log files.
logical block	A unit of data as it is handled by the software; the DG/UX system handles data in 512-byte logical blocks.
logical disk	A virtual disk. One or more formatted areas of a physical disk. Can be composed of up to eight pieces and can be distributed over several physical disks.
login	The program that controls logging in.
memory	Physical memory represents the available space in main memory. Programs are either swapped or paged

	into physical memory for execution. Virtual memory management techniques permit programs to treat disk storage as an extension of main memory.
mount	To extend the directory hierarchy by associating the root of a file system with a directory entry in an already mounted file system.
network	The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals. Networking for computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include file transfer, remote login, remote execution.
node name	An up-to-six character name for the system; used as the official name of the machine in a network. The node name resides in the NODE parameter.
null device	A device that always yields *end-of-file* on reading and discards all data on writing.
object file	A file of machine language code and data; object files are produced from source programs by compilers and from other object files and libraries by the link editor; an object file that is ready to run is an executable file.
operating system	The program for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, the file system, removing this burden from user programs. All other software runs under the operating system. The DG/UX system is an operating system.
open file	The destination for input or output obtained by opening a file or creating a pipe. Open files are shared across **forks** and persist across executions of **exec**. An open file need not exist in a file system, and a file may be opened by several processes simultaneously.
owner	The user ID of the process that created a file; the owner has distinctive permissions for a file.
page	A fixed length, 2048-byte block that has a virtual address, and that can be transferred between main

	and secondary storage.
paging	The process by which programs are truncated into *pages* and transferred between main and secondary storage by the virtual handler (or paging daemon).
parent process	See **fork**
parse	To parse is to analyze a sentence in order to identify its components and to determine their grammatical relationship. The meaning is similar in computer terminology, but instead of sentences, program statements or commands are analyzed.
pathname	A chain of names designating a file; a relative pathname leads from the current directory, for example, a path to directory A, thence to directory B, thence to file C is denoted A/B/C; a full pathname begins at the root, indicated by an initial '/', as in /A/B/C.
permission	A right to access a file in a particular way; read, write, execute (or look up in, if a directory); permissions are granted separately to owner, group, and others.
physical block	A unit of data as it is actually stored and manipulated; the DG/UX System handles data in 512-byte physical blocks.
pipe	A direct stream connection between processes, whereby data written on an open file in one process becomes available for reading in another.
pipeline	A sequence of programs connected by pipes.
polling	The interrogation of devices by the operating system to avoid contention, determine operation status, or ascertain readiness to send or receive data.
port	The point of physical connection between a peripheral device (such as a terminal or a printer) and the device controller (ports board), which is part of the computer hardware.
portability	The degree of ease with which a program or library can be moved (or ported) from one system to another. Portability is desired because once a program is developed, it can be used on many

	systems.
preprocessor	A generic name for a program that prepares an input file for another program. For example, **tbl(1)** is a preprocessor for **nroff(1)** and **cpp(1)** is a preprocessor for the C compiler.
process	A program in execution. It is an active entity capable of causing events to happen. A process is characterized by a core image with instruction location counter, current directory, a set of open files, control terminal, user ID, and group ID.
process ID	A unique, system-wide identification number for an active process.
profile	An optional shell script, **.profile** or **.login** used by the shell upon logging in to establish the environment and other working conditions of a particular user.
PSA	Primary system area; a formatted section of a physical disk created by the **diskman(1M)** program. The PSA contains the initial bootstrap program and information on the layout of the physical disk.
routine	A routine is a discrete section of a program written to accomplish a set of related tasks.
raw device	A character device, read and write operations to which are not buffered, and are synchronized to natural records of the physical device.
respawn	An action of the **init(1M)** program: to restart a process every time that process concludes.
root (/)	The directory that constitutes the origin of the directory hierarchy in a DG/UX file system; specifically, the origin for the file system with the conventional pathname '/'.
rotational gap	The gap between the actual disk locations of blocks of data belonging to the same file; the rotational gap compensates for the continuous, high-speed rotation of the disk so that when the controller is ready to reference the next physical block the read-write head is positioned correctly at the beginning of that block.
scheduler	A permanent process, with process ID 1, and associated kernel facilities that do scheduling.

secondary name server	A host that is configured to take over domain name server responsibilities temporarily in case the primary name server goes down.
sector	A 512-byte portion of a track which can be accessed by the magnetic disk heads in the course of a predetermined rotational displacement of the storage device.
segment	A contiguous range of the address space of a process with consistent store access capabilities; the four segments are (i) the text segment, occupied by executable code, (ii) the **data segment**, occupied by static data that is specifically initialized, (iii) the BSS segment, occupied by static data that is initialized by default to zero values, and (iv) the stack segment, occupied by automatic data, see stack; sometimes (ii), (iii), and (iv) are collectively called data segments.
semaphore	An IPC facility which allows two or more processes to be synchronized.
setuid	A special permission for an executable file that causes a process executing it to have the access rights of the owner of the file; the owner's user ID becomes the effective user ID of the process, distinguished from the real user ID under which the process began.
setuid bit	The associated permission bit.
shared memory	An interprocess communication facility (IPC) which allows two or more processes to share the same data space.
shell	The program **sh(1)**, which causes other programs to be executed on command. The shell is started when a user logs in. Also refers to any other initial login program such as **csh(1)**.
signal	An exceptional occurrence that causes a process to terminate or divert from the normal flow of control.
single-user	A state of the operating system in which only one user is supported.
source file	The uncompiled version of a program or the unprocessed version of a file.
special file	An inode that designates a device, further categorized

	as either (i) a block special file describing a block device, or (ii) a character special file describing a character device.
spool	An acronym for simultaneous peripheral output online. To collect and serialize output from multiple processes competing for a single output service.
stack	A segment of the address space into which automatic data and subroutine linkage information is allocated in last-in-first-out fashion; the stack occupies the largest data addresses and grows downward towards static data.
standard error	One of three files described below under **standard output**.
standard input	The second of three files described below under **standard output**.
standard output	Open files, customarily available when a process begins, with file descriptors 0, 1, 2 and stdio names "stdin", "stdout", "stderr"; where possible, utilities by default read from the standard input, write on the standard output, and place error comments on the standard error file. Initially, all three of these files default to your terminal.
sticky bit	A permission flag that identifies a file as a sticky file.
sticky file	A special permission for a shared text file that causes a copy of the text.segment to be retained in the swap area to improve system response.
stream	A full duplex, processing and data transfer path in the kernel. It implements a connection between a driver in kernel space and a process in user space, providing a general character I/O interface for the user processes.
superblock	A block which describes the layout of a file system. There are two copies per file system, at the first and last block.
superuser	User ID 0, which can access any file regardless of permissions and can perform certain privileged system calls.
swap	To move the core image of an executing program

	between main and secondary storage to make room for other processes.
swap area	The part of secondary store to which core images are swapped; the swap area is disjointed from the file system.
symbolic link	An inode that contains the pathname of another. References to the symbolic link become references to the named inode.
symbol table	Information in an object file about the names of data and functions in that file; the symbol table and address relocation information are used by the link editor to compile object files and by debuggers.
system calls	The set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. System calls are invoked by user processes for system-dependent functions, such as I/O, process creation, etc.
system console	On workstations or general multiuser systems, this is the directly connected terminal used for communication between the administrator and the computer.
system name	An up-to-six character name for the system; resides in the SYS parameter.
table	An array of data each item of which may be uniquely identified by means of one or more arguments.
track	An addressable ring of sections on a disk or diskette. Each disk or diskette has a predefined number of concentric tracks, which allows the disk head to properly access sections of data.
trap	A method of detecting and interpreting certain hardware and software conditions via software. A trap is set to catch a signal (or interrupt), and determine what course of action to take.
tunable parameters	Variables used to set the sizes and thresholds of the various control structures of the operating system.
tuning	The reconfiguration of the operating system to incorporate the modifications into an executable

	version of the system or to improve system performance.
user ID	An integer value, usually associated with a login name; the user ID of a process becomes the owner of files created by the process and descendent (**forked**) processes.

End of Appendix

Index

Note: Boldfaced page numbers (e.g., **1-5**) indicate definitions of terms or other key information.

Installing and Managing the DG/UX™ System

093-701052-01

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

093-701052-01