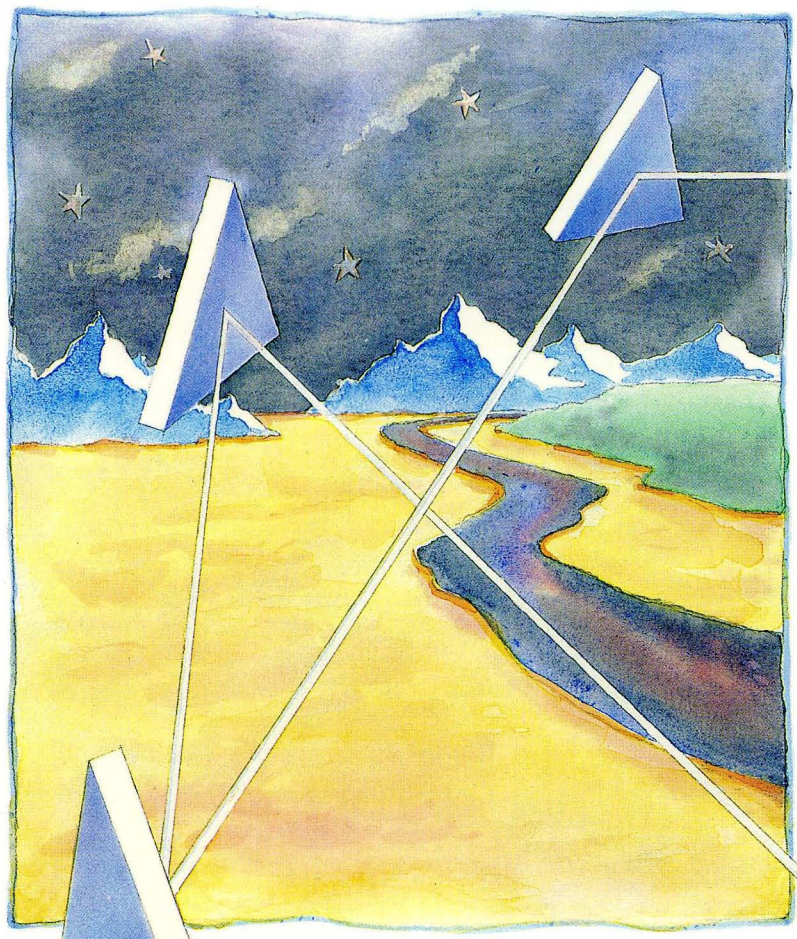


Managing BSD System Software

010853-A00

apollo

Managing BSD System Software

Order No. 010853-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: June 1988

This document was produced using the Interleaf Technical Publishing Software (TPS) and the InterCAP Illustrator I Technical Illustrating System, a product of InterCAP Graphics Systems Corporation. Interleaf and TPS are trademarks of Interleaf, Inc.

Preface

Managing BSD System Software describes system administration in the BSD environment. We've organized this manual as follows:

Chapter 1	Is an introduction to system administration in the BSD environment and a description of changes in Software Release 10 (SR10).
Chapter 2	Describes how to maintain nodes and provide services, including procedures to catalog nodes and manage root directories with **ns_helper**.
Chapter 3	Is a comprehensive discussion of the network environment, including both start–up procedures and files and server reference information.
Chapter 4	Describes registries, the **/etc/passwd** and **/etc/group** files, and how to update and replicate registry information.
Chapter 5	Discusses system and software security, the ACL system and Domain®/OS modes and how they interact.
Chapter 6	Describes the line printer system and how to configure and manage it in the BSD environment.
Chapter 7	Describes how to configure and maintain the **uucp** subsystem in the BSD environment.
Chapter 8	Documents **sendmail** and how it operates in the BSD environment.
Chapter 9	Contains the manual pages for BSD commands used for system administration.
Appendix A	Contains information on using **netmain** and **netmain_srvr**.

Intended Audience

This manual is intended for users familiar with BSD software, the Domain/OS system, and the UNIX* operating system.

The best introduction to the Domain/OS system is *Getting Started With Domain/OS* (Order No. 002348). This manual explains how to use the keyboard, display, read, and edit text, and manipulate files. It also shows how to request Domain/OS services using interactive commands.

If you are not familiar with the UNIX operating system, we recommend that you read one of the following documents:

- Bourne, Stephen W. *The UNIX System*. Reading: Addison-Wesley, 1982.
- Kernighan, Brian W. and Rob Pike. *The UNIX Programming Environment*, Englewood Cliffs, Prentice-Hall, 1984.
- Thomas, Rebecca and Jean Yates. *A User Guide to the UNIX System*. Berkeley: Osborne/McGraw-Hill, 1982.

Related Manuals

Making the Transition to SR10 Operating System Releases (Order No. 011435) describes how to make the transition from Software Release 9.7 (SR9.7) of the Domain operating system to Software Release 10. It includes an overview of new features and discusses the implications of operating a network of mixed-release (SR9.x and SR10) machines.

Managing Domain Routing and Domain/OS in an Internet (Order No. 005694) describes managing Domain/OS software in an internet environment.

Using Your BSD Environment (Order No. 0011020) is the first volume you should read. It explains how BSD works, and contains extensive material on the Bourne shell, C shell, and Mail.

BSD Command Reference (Order No. 005800) describes all the shell commands supported by BSD.

BSD Programmer's Reference (Order No. 005801) describes all the BSD system calls and library functions.

The *Domain C Language Reference* (Order No. 002093) describes C program development on the DOMAIN system. It lists the features of C, describes the C library, and gives information about compiling, binding, and executing C programs.

*UNIX is a registered trademark of AT & T in the USA and other countries.

The *Domain/OS Call Reference, Volumes 1 and 2* (Order Nos. 007196, 012888) describes calls to operating system components that are accessible to user programs.

Installing Software with Apollo's Release and Installation Tools (Order No. 008860) provides instructions on installing software on your system.

The *DPSS/Mail™ User's Guide* (Order No. 003660) describes the DPSS/Mail system and how to use it.

The *Domain Display Manager Command Reference* (Order No. 011418) provides descriptions of all the commands used for operating the DM.

Using TCP/IP Network Applications (Order No. 008667) provides instructions for using TCP/IP.

Configuring and Managing TCP/IP (Order No. 008543) describes how to configure and maintain TCP/IP on your network.

Programming with Domain/OS Calls (Order No. 005506) gives examples on how to use the Domain/OS calls.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo® Problem Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel, you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the **mkapr** (make apollo problem report) shell command description. You can view the same description online by typing:

$ man mkapr (in the SysV environment)

% man mkapr (in the BSD environment)

$ help mkapr (in the Aegis environment)

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about the manual.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.
user-supplied values	*Italic words or characters in formats and command descriptions represent values that you must supply.*
sample user input	In examples, information that the user enters appears in color.
`output`	`Information that the system displays appears in this typeface.`
[ ]	Square brackets enclose optional items in formats and command descriptions.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on the keyboard.
CTRL/	The notation CTRL/ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.
. . .	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.
. . .	Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.
——— ⊞ ———	This symbol indicates the end of a chapter.

Contents

Chapter 3 Administering Nodes in the Network

Chapter 4 Creating and Maintaining the Registry

Chapter 5 Protection of Files and Directories

Chapter 6 Line Printer Management

Chapter 7 uucp Administration

Chapter 8 sendmail Configuration and Usage

Contents 9 Administrative Commands

Appendix A Using the netmain Interactive Tool and netmain_srvr in the Apollo Token Ring Network

Illustrations

Tables

Chapter 1

An Overview of BSD System Administration

Contents

Chapter 1
An Overview of BSD System Administration

This manual provides an overview of system administration in the BSD environment, as well as a summary of the major changes made to the system at Software Release (SR10). The manual refers to all Apollo devices that communicate on the network as "nodes," although, where appropriate, we distinguish between nodes with displays and keyboards (for example, the DN3000 series) and nodes without (for example, the DSP160) since they are sometimes configured in different ways.

Information in this manual also assumes a single Domain network; if your site operates in an internet, please refer to the book *Managing Domain Routing and Domain/OS in an Internet* for any additional information about system administration in that environment.

This book deals with system software, that class of programs that manages the functioning of the operating system. We assume a certain level of familiarity with BSD user-level commands and concepts. If you've read and understood the *Using Your BSD Environment*, you should have no difficulties with anything explained in this book.

1.1 System Administration Responsibilities

As a system administrator, you are generally responsible for some or all of the following tasks:

- Enabling nodes to communicate in the network by cataloging disked nodes, providing partners for diskless nodes, and maintaining root directories.
- Creating processes to provide both network-wide and per-node services like printing, remote login, and diskless node booting.
- Understanding how to configure and administer individual nodes within the network. (While individual node users will sometimes wish to determine the system software that runs on personal nodes, the system administrator is almost always a source for advice and assistance.)
- Creating and managing a registry of authorized user and account information, including accounts and group and organization lists.
- Determining access to system software, programs, and users' files.
- Backing up both system software and user files on a regular basis.

1.2 Changes to the Operating System at SR10

We've made some major changes to broad areas of the operating system at this release. The major areas of the operating system affected, especially with regard to system administration, are the registry, protection (file and directory permissions), and the way in which the names of file system objects are represented. In the following subsections, we provide a brief discussion of the changes and their impact on BSD system administration. Complete information about these areas, as well as procedures to perform common administrative tasks, can be found in the appropriate chapters of this book. For information on conversion tools and compatibility between pre-SR10 and SR10 versions of these subjects, see *Making the Transition to SR10 Operating System Releases.*

1.2.1 Case Sensitivity and Names

At SR10, the operating system is completely case sensitive. We do provide conversion tools, however, to ease the transition from a case-insensitive environment to a case-sensitive one. See *Making the Transition to SR10 Operating System Releaes.*

The most obvious effect of case sensitivity in the operating system is that the system will interpret mixed-case pathnames literally. For example, the pathnames **/DIRECTORY/FILENAME**, **/directory/filename**, and **/DiReCtOrY/fIlEnAmE** are no longer equivalent by default.

If you don't need to use mixed-case names, you can avoid many problems by setting the environment variable DOWNCASE to TRUE. From the point of view of naming, this will give you a pre-SR10 environment. However, DOWNCASE is intended as a temporary measure and may become obsolete at a future release. Don't rely on this mechanism for the long term.

Case sensitivity can also introduce some incompatibilities in existing shell scripts and programs. For a full discussion of the impact of the changes to naming at SR10 on programs and shell scripts, see *Making the Transition to SR10 Operating System Releases.*

The Naming Server Helper (**ns_helper**) will continue to be case insensitive, but we recommend to system administrators that new node names reflect the case-sensitive world, both to avoid confusion and because **ns_helper** may become case sensitive at a future release. The **ns_helper** is discussed in Chapter 2.

The maximum number of characters in a leaf (single file or directory) name has been increased from 32 characters to 255 characters. The maximum length of a pathname that the system can handle has increased from 256 characters to 1023 characters.

1.2.2 The Registry

The registry is the mechanism that controls user access and authentication; it has undergone significant change at SR10. Registry information is now stored in a replicated database which is managed by servers based on the Apollo Network Computing System™ (NCS).

A registry server and registry server database manage the overall registry function, and the operating system gains access to registry information via the registry server. Each node has a local registry available to provide node-specific registry history information so that a user

can log in in the event of network failure. A local cache of name-to-UID (Unique Identifier) mappings is maintained on each node to improve performance.

The SR10 registry includes the concepts of membership lists, groups and organizations, which allows sites some flexibility in how the registry information is maintained. It also introduces the concept of ownership as a means of controlling access to registry database information. Simply stated, you must own a registry database relation to be able to change it. With these two additions, it is now an easy matter to partition a network's registries into logical groupings of organizations and groups, simplifying system administration.

The system administrator manipulates accounts by means of the **edrgy** tool. With **edrgy**, you can create and delete accounts, as well as edit and perform global operations on other registry database information.

It is possible to run a mixed network of pre-SR10 and SR10 machines, but you'll probably wish to site the SR10 registries and the pre-SR10 registries on different nodes. If your network is small enough that keeping two types of registries will absorb too much disk space or be too confusing otherwise, you should consider converting to SR10 all at once. Information about operating in an environment of mixed registries is available in *Making the Transition to SR10 Operating System Releases.*

Complete information about creating and maintaining SR10 registries is available in Chapter 4. Tools are available to convert existing registries from pre-SR10 to the SR10 format and to convert SR10 registry information back to the pre-SR10 form. The intention is that, at some future release, all registries will be converted to the SR10 format. Descriptions of the various registry conversion tools and procedures can be found in *Making the Transition to SR10 Operating System Releases.*

1.2.3 Protection: The Access Control List

At SR10, the Access Control List (ACL) mechanism, which manages file system object protection, has been simplified and altered. Every object in the file system has an ACL that consists of four required entries for owner, group, organization, and world. Each entry consists of a Subject Identifier (SID) and some rights specifications. Additional protection entries, if required, are stored in an "extended ACL" that is essentially like the pre-SR10 ACL. See Chapter 5 for information about ACLs.

As a result of these changes, so-called "canned" ACLs, predefined sets of rights specifications for certain account names, are no longer supported. The changes to ACLs also include new versions of the **acl**, **edacl**, and **salacl** commands to operate with the new and changed rights. Note that the new ACL structures will also have an impact on what protection information is preserved on backups.

Because of the general incompatibility between the SR10 ACL manager and the ACL manager prior to SR9.7, there is no way to share files between pre-SR9.7 and SR10 nodes. If your site has few enough nodes that you can update to SR10 all at once, you should do that; if your site is upgrading to SR10 over a long interval, and you must be able to access all files on all nodes at all times, you should install SR9.7 before updating any nodes to SR10. For information about transition information and system software installation, see the books *Making the Transition to SR10 Operating System Releases* and *Installing Software with Apollo's Release and Installation Tools*.

At SR10, all mapping between UNIX modes and ACLs is handled transparently, without any intervention by the system administrator or user. The entire UNIX protection model operates exactly as you would expect in a "standard" UNIX system.

Chapter 2

Maintaining Nodes and Providing Services in the Network

Contents

Chapter 2
Maintaining Nodes and Providing Services in the Network

This chapter assumes that your installation consists of a single Domain network. If you have multiple Domain networks connected in an internet, you have other considerations. See *Managing Domain Routing and Domain/OS in an Internet* for additional information about how to catalog nodes and maintain root directories in an internet environment.

Topics covered in this chapter include cataloging nodes and maintaining node root directories, using the **ns_helper** process to maintain root directories, and providing such network-wide services as printing and remote log in.

2.1 The Root Directory

To communicate over the network, node must recognize each other. Each node has a root directory that associates node names and hexadecimal node IDs for all the nodes on your network; this ensures that communication and file access between and among nodes can take place. You must maintain node root directories accurately if all the nodes in your network are to operate efficiently. If a node's root directory is incomplete or inaccurate, the node may be unable to communicate with other nodes on your network.

This section describes:

- Node names and node IDs
- The process of cataloging associations between node names and node IDs
- How to maintain root directories
- An automated way of maintaining nodes' root directories, the **ns_helper** (Naming Server Helper) process

2.1.1 Node IDs

Every node has a unique hexadecimal ID number (the node ID) which is contained in a Programmable Read-Only Memory (PROM). The only way a given node's ID changes is if a service representative physically replaces the node's ID PROM. The node ID allows both the network communications software and other nodes' software to recognize that node.

Node IDs in an internet have a network number as a prefix to the hexadecimal ID that identifies the individual machine. See *Managing Domain Routing and Domain/OS an Internet* for details.

2.1.2 Node Names

Since hexadecimal numbers are not easy to remember, you can associate a node name with a particular node ID and refer to the node by name when using shell commands like **lvolfs** (list volume free space) that allow you to specify a node with the **-n** option. All the name-ID associations that a node knows about are stored in the node's root directory.

A node name must begin with a letter, and all alphabetic characters in the name must be lowercase. You can assign node names to both disked and diskless nodes, but a diskless node's name does not always act the same way as a disked node's. In particular, you should remember that a diskless node's name is not the same as its entry directory name, as is the case with disked nodes. For example, if the node **dublin** were disked, the following command would list the contents of **dublin's** entry directory.

```
% ls //dublin
```

If the node **dublin** were diskless, the same command would result in "object not found."

You associate node names with node IDs by means of the **ctnode** (catalog node) command. Later in this chapter, you'll find procedures that demonstrate how to catalog nodes, both in the node's own root directory and in the root directories of other nodes.

2.1.3 Cataloging a Node

The **ctnode** command enters the node's name, hexadecimal ID, and other information in the root directory. You must catalog a node whenever you

- Add a new node to the network.
- Change a cataloged node's name.
- Replace a node's disk.
- Run the **invol** utility on a node's disk.
- Have a node's ID PROM replaced. The PROM installation procedures recatalog the node's directories with its new ID. You then must update root directories on the rest of the network with the new node ID.

A new disked node arrives at your site already cataloged in its own root directory. Its default name is *node_nnnnn*, where *nnnnn* is the node's hexadecimal ID number. Diskless nodes are not already cataloged. We strongly suggest that you name all the nodes in your network.

Cataloging a node is a two-step process. First you must catalog the node in its own root directory (or, for diskless nodes, in the partner's root directory). Then, you must make this information available to all other root directories in the network. How you propagate the node name information to the other root directories on the network depends on whether your network uses the **ns_helper** process to maintain root directories.

The **ns_helper** maintains a master copy of the root directory and provides node-name to node-ID associations. It reduces the cataloging effort when you add nodes or change names, and is very useful in larger networks. You must run the **ns_helper** process if you have a Domain internet, and you should run it if your network configuration changes frequently. For complete information and procedures for using **ns_helper** and the **edns** tool which accompanies it, see Section 2.3.

If your network is small, node names seldom change, and new nodes are added to the network infrequently, you probably don't need to run **ns_helper**. In this case, you'll use the **ctnode** and **uctnode** (uncatalog node) commands and Procedures 2-2 through 2-6 in this chapter to maintain the root directories of nodes on your network. Many of these procedures will not operate in an internet. See *Managing Domain Routing and Domain/OS in an Internet* for information about using **ns_helper** on a Domain internet.

2.1.4 Cataloging a Node in its Own Root Directories

Use Procedures 2-1 and 2-2 to catalog a disked or diskless node in its local root directory. The procedures catalog the diskless node in its partner node's root directory. Use one of these procedures whenever you catalog a node except for when a PROM is changed. In this case, the PROM installation procedures recatalog the node in its own root directory; however, you must still recatalog the node in other nodes' root directories if you do not use **ns_helper.**

- Use Procedure 2-1 to catalog a node that has a display (that is, any node except a Domain Server Processor).
- Use Procedure 2-2 for a a server node that does not have a display.
- If you are cataloging more than one node, use Procedure 2-1 or 2-2 at each node you are cataloging.
- These procedures only catalog a node in its local root directory. If you do not use **ns_helper**, you must continue with Procedure 2-3, 2-4, 2-5, or 2-6 to provide this information to all other nodes.

Please read through each procedure before you attempt to carry it out. If you receive error messages when you carry out the procedures, check the command line to be certain that you have given the correct input. If you are sure you are giving the correct input but you continue to receive error messages, check with your designated service representative.

PROCEDURE 2–1. *Cataloging a Node in its Own Root Directory*

Task 1: Log in as user

Task 2: Determine the node's hexadecimal ID

```
% lcnode -me
The node ID of this node is 8523.
```

Task 3: Uncatalog the old node name

If this is a new diskless node, you replaced the disk on an already–cataloged node, or you ran **invol**, go to Task 4. If this is a new disked node, the initial node name is the node ID preceded by **node_**, for example, node_8523. In the following example, the **–l** option lists the node's name after it is uncataloged.

```
% uctnode node_8523 -l
 "node_8523" uncataloged.
```

Task 4: Catalog the new node name

Enter the following command if you are cataloging a new node or are giving a node a name that has never been used before. For example, to name the node with hexadecimal ID 8523 "salmo," type the following.

```
% ctnode salmo 8523 -l
  Node 8523 cataloged as "salmo".
```

Enter the following command if you are reusing an existing name. You usually reuse a name when you change disks, run **invol**, or replace a node and the user wishes to keep the old node name for the new node.

% **ctnode** *old_name node_id* **–l –r**

Task 5: Update the node's root directory

This step adds node name–ID associations for other nodes on the network to this node's master root directory.

```
% ctnode -update -l
 3 nodes responded!
 Node 8555 cataloged as "arctic_char"
 Node 8523 cataloged as "rainbow"
 Node 8525 cataloged as "brook"
```

PROCEDURE 2–2. *Cataloging a Domain Server Processor*

Use this procedure to catalog Domain Server Processors (DSPs). If you're setting up a new network, catalog DSPs after you've cataloged nodes with monitors. Get the DSP's node ID from the inspection slip attached to the shipping carton packing slip. If you do not have the inspection slip, contact your service representative; this is the only reliable way to determine the node ID when the node is uncataloged and you don't have the packing slip. You must have the node ID before you start this procedure.

Task 1: Log in to the DSP as user

Enter the following command at a shell prompt on a node with a monitor. Note the two single quotes ('') at the end of the command line, which show that the account user has a null password. For example, if the DSP's node ID is 8533, type the following:

```
% crp -on 8533 -login user ''
Connected to node 8533
```

Task 2: Uncatalog the old node name

If you replaced the disk on an already–cataloged node, or you ran **invol**, go to Task 3. If this is a new DSP, the initial node name is the node ID preceded by **node_**, for example, node_8533. In the following example, the **–l** option lists the node's name after it is uncataloged.

```
% uctnode node_8533 -l
 "node_8533" uncataloged.
```

Task 3: Catalog the new node name

Enter the following command if you are cataloging a new DSP or are giving a DSP a name that has never been used before. For example, type the following to associate the name "chinook" with the DSP with node ID 8533.

```
% ctnode chinook 8533 -l
Node 8533 cataloged as "chinook".
```

Enter the following command if you are reusing an existing name. You usually reuse a name when you change disks, run **invol**, or replace a node and the user wishes to keep the old node name for the new node.

% ctnode *old_name node_id* –l –r

Task 4: Update the node's root directory

This step adds node name—ID associations for other nodes on the network to this node's master root directory.

```
% ctnode -update -l
 3 nodes responded!
Node 8523 cataloged as "rainbow"
Node 8525 cataloged as "brook"
Node 8533 cataloged as "chinook"
```

2.2 Using ctnode to Catalog Nodes on the Network

Once you catalog a node in its own root directory, you must then provide the information to all other nodes' root directories so that remote nodes can communicate with the newly cataloged node and have access to its files. If the network is small and node configurations don't change often, you can use the **ctnode** and **uctnode** commands to manage the network root directories. Procedures 2–3 through 2–6 show the steps you must follow to update the root directories. Use these procedures as detailed below. If you have a larger network, you should probably run the **ns_helper** process. Go to Section 2.3 for information and procedures for using **ns_helper**.

- Use Procedure 2–3 to create a new network or to add several nodes to a network.
- Use Procedure 2–4 to add a single node to an existing network.
- Use Procedure 2–5 to change the name of a node that is already on the network.
- Use Procedure 2–6 after replacing a disk drive, running **invol**, or if your service representative replaced a node's PROM.

All these procedures assume that you've already cataloged the node in its own root directory, using either Procedure 2–1 or 2–2.

PROCEDURE 2–3. *Creating a New Network*

Task 1: Log in as user

Task 2: Update the root directory

Enter the **ctnode -update** command to update the node's root directory to include information on all nodes that are currently responding to network queries. In the following example, the **-l** option lists the nodes as they are cataloged.

```
% ctnode -update -l
2 nodes responded!
Node 8555 cataloged as "arctic_char"
Node 8525 cataloged as "brook"
```

The local node now has a complete root directory. If the number of nodes responding does not equal the number of nodes in your network, repeat Task 2 until you get a full root directory.

Task 3: Propagate new information across the network

You must propagate the new name–ID information to the root directories of all other nodes. Enter the name of the node you're logged into in place of *//node_name* in the following command line.

% ctnode -md -from *//node_name* **-on //?***

PROCEDURE 2–4. *Cataloging a New Node in an Existing Network*

Task 1: Log in as user

Task 2: Catalog the new node

Catalog the new node on all other nodes in the network with the following command. Substitute the node's name and ID in the appropriate places.

% ctnode *node_name node_ID* **-on //?***

PROCEDURE 2–5. *Changing a Node's Name*

Task 1: Uncatalog the old name

For each node on the network for which you want to uncatalog the node's old name, log in as **user** and enter the **uctnode** command to remove the node's old name from the root directory, as follows:

```
% uctnode cutthroat -l
 "cutthroat" uncataloged.
```

If you do not perform this step, the node will be cataloged under both the new and the old name.

Task 2: If you are not still logged in, log in to any node as user

Task 3: Update root directories

To propagate the new node name to the root directories of all nodes in the network, recatalog the node under its new name. In this example, the node ID is **cff**.

```
% ctnode sockeye cff -r -on //?*
```

PROCEDURE 2–6. *Updating Information for an Existing Node Name*

Use this procedure after replacing a disk drive, running **invol**, or if your service representative replaces a node's PROM.

Task 1: Log in to any node as user

Task 2: To recatalog the node in its own root directory, substitute the node's name and ID in the following command:

% ctnode *node_name node_ID* -r

Task 3: Update the root directories across the network

To propagate the updated information into the root directories of all nodes in the network, enter the recataloged node's name and ID in the following command:

% ctnode *node_name node_ID* -r -on //?*

2.3 The Naming Server Helper: ns_helper

The **ns_helper**, the Naming Server Helper, is a Domain server process that provides an automated method of maintaining node root directories. You can run **ns_helper** on any disked node in Domain network. You must use it on each Domain network in an internet. See *Managing Domain Routing and Domain/OS in an Internet* for information about running **ns_helper** in an internet.

The **ns_helper** (**/sys/ns/ns_helper**) process manages a master root directory. This database is the only comprehensive source of node identifying information in the network. The **ns_helper** performs most of its operations automatically. The **edns** utility, an interactive tool used with **ns_helper**, is available for those operations requiring your intervention such as updating the database.

The **ns_helper** server maintains a cache of the master network root directory at each node. Whenever the naming server uses the master root directory to locate objects, it updates the local node's cache. Although the shell command **ctnode** is operative, you need not maintain a node's root directory with **ctnode -update** in the **ns_helper** environment. It is always necessary to catalog an entry directory name with **ctnode** when a node is first brought into the network.

When more than one **ns_helper** runs in a network, each process is called a replica. The The **ns_helper** server propagates changes in the database of any replica to all other replicas for a period of 14 days. In exceptional circumstances of node, loop, or disk failure, a replica may not receive updated information in this time period. Use an **edns** merge command to return replicated databases to a consistent state in these cases.

We recommend running **ns_helper** as a background process. Enable **ns_helper** from the appropriate start-up file so that it will continue after logout. The **ns_helper** server names itself "ns_helper" by default, so you need not specify the **-n** option to the process creation command.

2.3.1 The ns_helper Database

The **ns_helper** manages a database that is divided into two parts: a master root directory and a replica list. The master root directory is the comprehensive source of node identification information in the network. You can specify the node names and addresses in the master root directory. Only the nodes themselves can supply **ns_helper** with the rest of the information in the directory. The **ns_helper** database resides in the **'node_data/system_logs** directory.

On large networks and on Domain internets you can have more than one **ns_helper** process, each with its own copy of the database; these are called replicated **ns_helpers** and databases. In this case, the database replica list includes the nodes that run **ns_helper**.

Table 2-1 lists the **ns_helper** database contents in detail.

Table 2–1. Contents of the ns_helper Database

Item	Definition
	Master Root Directory
Node Name	The name of the node.
Address	The network number (0 for Domain single networks) and the node's hexadecimal ID.
Entry Type	Type of object — for disked nodes, system directory (sdir); for diskless nodes, node (node).
UID	The Unique Identifier (UID) for a node's entry directory For diskless nodes, **ns_helper** assigns a UID.
Entry Date and Time	The date and time at which this entry was added to the master root directory.
Creating Node	The hexadecimal ID of the node at which the entry was added to the master root directory.
	Replica List
Replica List	The hexadecimal IDs of all nodes that run the **ns_helper** process.

2.3.2 When to Use ns_helper in Your Network

Use **ns_helper** in networks where new nodes are introduced frequently, when many nodes have multiple users, and when you want to maintain all the node root directories from one location. As we said before, you must run **ns_helper** on each network in an internet.

2.3.3 Number and Placement of Replicated ns_helpers

In smaller networks, a single **ns_helper** process provides a reliable way to keep nodes up to date. In some environments, however, you may choose to run the **ns_helper** server on several nodes. Consider running more than one **ns_helper** process when

- You have many nodes in your network, and a single server may not be able to handle the traffic.
- You want to ensure that the server process is always available; two or more servers will provide redundancy.
- Loops in your network are switched out regularly and/or your network runs through several buildings. You might want servers in each loop or building.

You can run **ns_helper** only on disked nodes.

2.3.4 Replica List

When more than one **ns_helper** runs on a network, the server processes maintain information about each other in a part of their database called the replica list (see Table 2–1). The replica list contains the hexadecimal ID of every node running **ns_helper**. You manage replicated **ns_helpers** with the **edns** command.

Each **ns_helper** automatically tries to keep its own database (master root directory and replica list) consistent with those of the other **ns_helpers**. When you make changes to any database, that node's **ns_helper** refers to its replica list for the node IDs of other replicas. Then the **ns_helper** sends the new information to all the other nodes on the list. When **ns_helpers** receive new information from another server, they update their own databases and return an acknowledgment to the sending server.

If the sending **ns_helper** does not receive an acknowledgment from all the nodes on its replica list, it continues to propagate the new information for a few days. In exceptional circumstances, a replica might never receive updated information. You can use the **edns merge** facility to return replica databases to a consistent state in these cases.

2.3.5 Managing Root Directories with ns_helper

When **ns_helper** runs in a network, the naming server (the part of the operating system that locates file–system objects) has two sources of information about entry directory names: the node's local root directory and the **ns_helper** master root directory itself.

When the naming server tries to locate an object, it first looks in the node's root directory. If the name isn't there, the naming server refers to **ns_helper**'s master root directory for information about the entry name. Whenever the naming server gets information from the master root directory, it adds that information to the node's root directory.

Using ctnode and uctnode with ns_helper

When you use **ns_helper** on the network, you normally will not need to use the **ctnode** command to maintain a node's root directory.

There is one situation in which you'll need to use **uctnode:** When you change the name of a node in the **ns_helper** database, the new name is added to the individual nodes' root directories, but the old name is not deleted from any of them. Use **uctnode** at each node on the network to remove the old entry from all root directories.

> **NOTE:** After **uctnode** removes an entry directory name, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

The following subsections describes tools, considerations, and procedures for managing root directories on networks that use **ns_helper**.

The edns Utility

The **edns** (edit naming server) utility manages the **ns_helper** and its database. Table 2–2 briefly describes the **edns** commands. The *BSD Command Reference* describes the **edns** commands in greater detail. Online help is also available by using the **help** command.

Table 2–2. The edns Commands

Command	Description
add	Adds a node name and the corresponding address to the master root directory(s).
addrep	Adds the address of an **ns_helper** node to the **ns_helper** replica lists(s).
cmp	Compares two **ns_helper** databases and lists duplicate or inconsistent entries.
delete	Deletes the entry for the specified name from the **ns_helper** master root directory(s).
delrep	Deletes an **ns_helper** node from the **ns_helper** replica lists.
diff	Lists differences between two **ns_helper** databases, including the master root directories and replica lists.
info	Displays address and status information for the default **ns_helper**.
init	Initializes an **ns_helper** database with data from all nodes that are currently responding on the network.
ld	Lists master root directory information.
lr	Lists addresses of all **ns_helper** nodes on the network and, optionally, their current clock dates and times.
merge	Merges all entries from one **ns_helper** master root directory (but not replica list) into another.
merge_all	Performs a global merge of all **ns_helper** databases, using the default or specified **ns_helper** database.
quit	Ends the current **edns** session.
replace	Changes the address and UID associated with the specified name.
set	Sets the default **ns_helper** to the one running on the specified node.
shut	Shuts down the **ns_helper** on the specified node. This command deletes that node's database but does not remove the node from other **ns_helper** replica lists.
update	Updates all replica master root directories with data from all nodes that are currently responding on the local network.

Synchronizing Clocks on Replicated Databases

The **ns_helper** processes keep only the most recent information about an entry in their databases. The servers use the node hardware clocks and the database item "Entry Date/Time" to recognize the most recent information. Therefore, you must keep the hardware clocks on all **ns_helper** nodes synchronized. Check the node clocks periodically and reset them if they diverge by more than a few minutes. The **edns** command provides a way to check clock synchronization (see Procedure 2–15). (Procedure 2–7 explains how to synchronize node clocks.)

Network Availability and edns

Two **edns** operations, **initialize** and **update**, are particularly sensitive to network and node availability because they request information from all nodes on the network. If a node fails to respond, it may not be cataloged in the root directory. Therefore, it is a good idea to initialize the first **ns_helper** process at a time when network traffic is light and all nodes are connected to the network.

The edns Utility and Diskless Nodes

When **edns** initializes a database, it always assigns the default name

diskless_$*nnnnnn*

to a diskless node, where *nnnnnn* represents the diskless node's hexadecimal ID. The value is right justified and is preceded by the number of zeros required to form a six–digit number. For example, if the node ID is 3d3, the **edns** representation of that node is

diskless_$0003d3

Edns generates a Unique Identifier (UID) for a diskless node, then associates it with the diskless node's name. This information about the diskless node appears in the master root directory and can be copied to a node's root directory. In a single Domain network, the UID and other information that **edns** generates for the diskless node serves only as a place marker for information that is used in a Domain internet.

To name a new diskless node on an existing network that runs an **ns_helper** process, use the **ctnode –root** command described in the next section, or use Procedure 2–10 described in Subsection 2.3.7. If the node was on the network when **ns_helper** was initialized, it already has a default name; in this case, use Procedure 2–12 or use **uctnode –root** to uncatalog the node before recataloging it.

The command **ls // –ln** lists the names of all diskless nodes in the local copy of the root directory. The following commands also specify if a node is diskless. (The *node_spec* argument can be either a hexadecimal node ID or the node entry directory, that is, **//node_name**.)

% **netstat –n** *node_spec*
% **lusr –n** *node_spec*
% **bldt –n** *node_spec*
% **lcnode –me**

2.3.6 User Procedures for Updating the Master Root Directory

The **ctnode** and **uctnode** commands support a small subset of operations on the master root directory. Any user can run these commands to manage the master root directory entries.

Use the following command to delete the entry for a node name in both the node's root directory and the master root directory. If you remove a node from the network, this command removes the old node's entry from the master root directory. If you are changing a node's name, use this command to remove the entry for the old name before adding the new name.

% **uctnode** *node_name* **-root**

> **NOTE:** Any time you change a node's name, you must use **uctnode** on each node to delete the old entry from that node's root directory.

Use the following command to add a node name in the root directory and the master root directory. You can use this command to give a diskless node a name or to add a new node to the network.

% **ctnode** n*ode_name node_id* **-root**

Use the following command to replace the node ID or UID that is associated with an existing node name in the root directory and the master root directory. You can use this command if your disk is changed or if you run **invol**. You can also use this command to associate an existing name to a new node.

% **ctnode** *node_name node_id* **-root -r**

2.3.7 System Administrator Procedures for ns_helper

The rest of this chapter contains procedures for managing **ns_helper** processes and databases. Table 2–3 lists tasks you might wish to perform and gives the number of the appropriate procedure. In secure networks, we recommend that you set the ACLs so that only the system administrator (%.sys_admin.%) can use the **edns** command. If you do, only the system administrator can perform Procedures 2–9 through 2–16 that invoke **edns**. Other users must run the **ctnode** and **uctnode** commands to manage a node's entry in the master root directory as described in the preceding section.

Table 2–3. The ns_helper Procedures

Description of Procedure	Number
Add a node to an existing network	2–10 *
Add node names to the **ns_helper** database	2–10 *
Add an **ns_helper** replica	2–14
Change a node's name in the **ns_helper** database	2–12 *
Check clock synchronization on **ns_helper** nodes	2–15
Delete names from the **ns_helper** database	2–11 *
Give a diskless node a name	2–10, 2–12 *
Initialize a network's **ns_helper** database	2–9
Maintain the consistency of replicated **ns_helper** databases	2–16
Reinitialize a single **ns_helper** process	2–14
Remove an **ns_helper** replica	2–17
Remove a node from the network	2–11 *
Repair a replica	2–16
Start **ns_helper** on one or more nodes	2–8
Stop an **ns_helper** process	2–18
Synchronize node clocks	2–7
Update **ns_helper** after changing a PROM or running **invol**	2–13 *

NOTE: An asterisk (*) indicates a procedure that you can also perform by using the **ctnode** and **uctnode** commands.

PROCEDURE 2-7. *Synchronizing Node Hardware Clocks*

Use this procedure to synchronize node hardware clocks. You must use this procedure if nodes that run **ns_helper** are not within five minutes of each other. You should use the procedure if the **ns_helper** nodes are not within one minute of each other.

Task 1: Obtain an accurate timepiece

Task 2: Set the node's NORMAL/SERVICE switch to SERVICE

Task 3: Shut down the system

If the node is a DSP unit without a display, you must attach either a terminal or a Domain node with a display to the DSP unit's SIO (Serial Input/Output) line to set the node clock. Once you've done that, shut down the DSP.

If the node has a display, shut it down with the Display Manager (DM) **shut** command.

Task 4: Start the calendar program

The Mnemonic Debugger prompt (>) appears on the screen. Enter the following command:

> ex calendar

The **calendar** program prompts you for the required responses. (You must answer all questions; you cannot set the time without also entering the date.) For example, the following script illustrates prompts from **calendar**, and the responses for a node with a Winchester disk:

```
> ex calendar
Please enter disk type (W,S, or F)[,lvno].
If you do not have a disk, enter none (N): w

The time-zone is set to -4:00(EDT).
Would you like to reset it? n

The calendar date/time is 1988/07/11 13:52:03 EDT.
Would you like to reset it? y

Please enter today's date(year/month/day): 1988/07/11
Please enter the local time in 24 hr. format (hours minutes) 13:55

The calendar has been set to 1988/07/11 13:55 EST (1986/07/11
18:55:04 UTC)
```

NOTE: If you set a node clock backward, the node can generate duplicate UIDs for objects, which will create confusion in the operating system. You can avoid this by not rebooting until the amount of time you set the node clock past has elapsed. For example, if you set the clock back by one hour, wait one hour before you reboot the node.

Task 5: Reboot the node or DSP

Return the NORMAL/SERVICE switch to the NORMAL position and reboot. Type

```
> re
> <RETURN>
> ex domain_os
```

Task 6: Repeat Tasks 2 through 5

Using the same timepiece, repeat Tasks 2 through 5 at each of the nodes you selected to run **ns_helper**.

PROCEDURE 2-8. *Starting the ns_helper Server Process*

Use this procedure to start or restart the **ns_helper** on any node that maintains a master root directory.

Task 1: Check node clock synchronization

If more than one node runs (or will run) **ns_helpers**, check to make sure that the nodes' clocks are within one minute of each other. To do so, enter the **netstat -n** command, followed by the node specifications of the nodes that run **ns_helpers**. In this example, the nodes **brook** and **golden** run **ns_helper**.

```
% netstat -n //brook //golden
     The net_id.node_ID of this node is 0.5678.
**** Node 4567 ****  //brook
Time 1986/07/01.15:25:57  Up since 1986/07/01.14:38:25
Net I/O:          total=  24555   rcvs =  17930   xmits =   6625
Winchester I/O:   total=  13837   reads=  11098   writes=   2739
No ring hardware failure report.
System configured with 3.0 mb of memory.
**** Node 1345 ****  //golden
Time 1986/07/01.15:21:44  Up since 1986/06/24.20:57:25
Net I/O:          total= 5572914   rcvs = 3892617   xmits = 1680297
Winchester I/O:   total= 244976   reads= 148445   writes=  96531
System configured with 2.5 mb of memory.
```

If the times reported are not within one minute of each other, use Procedure 2-7 to synchronize the hardware clocks on the **ns_helper** nodes.

Task 2: Log in

Log in to the node that will run **ns_helper**. Use the appropriate procedures for nodes with monitors or DSPs.

Task 3: Edit the start-up file

Add the following line to the appropriate start-up file:

```
cps /sys/ns/ns_helper
```

This line will start an **ns_helper** process on this node, when the node is rebooted.

Task 4: Start the ns_helper process

If you are working at the node you want to run **ns_helper**, enter the following command in the DM input window:

```
Command: cps /sys/ns/ns_helper
```

If you are working at another node (for example, if the **ns_helper** node is a DSP server, enter the following command. You must use this command even if you are logged in remotely to the **ns_helper** node.

% crp -on *node_spec* **-cps /sys/ns/ns_helper**

Where *node_spec* is the name of the node.

Task 5: Verify that the server process is running

Enter the **pst** command. The output from the command is shown below.

```
% pst <RETURN>
------------------------------------------------------------
Processor  | PRIORITY | Program | State |  Process
Time (sec) | mn/cu/mx | Counter |       |  Name
------------------------------------------------------------
 70.938      16/16/16   1BDE6     Wait    display_manager
 21.297      1/14/16    <active>  Ready   process_7
  0.850      1/14/16    1BD46     Wait    ns_helper
.
.
.
```

Task 6: Repeat Tasks 2 through 5

Repeat Tasks 2 through 5 at each of the nodes you selected to run **ns_helper**.

PROCEDURE 2–9. *Initializing the Network ns_helper Database*

Use this procedure to initialize the **ns_helper** database on a new Domain network. We illustrate the following procedure with **ns_helper** running on two nodes: **golden** (ID 8521), and **brook** (ID 8525) and with six nodes in the network: **golden, brook, grayling, gila, arctic_char,** and **coho.**

Task 1: Catalog all nodes in their own root directories

Be sure that all nodes in the network have their own entry directory names cataloged in their own root directories. Procedures 2–1 and 2–2 show how to do this.

Task 2: Start ns_helper on each helper node

Use Procedure 2–8 to start the **ns_helper** process on the nodes you've selected.

Task 3: Start edns

Invoke **edns** from any node in the network with the node specification of the node that will run **ns_helper**. Our example selects **brook**; its **ns_helper** process becomes the default **ns_helper.**

```
% edns 8525
the default ns_helper is 0.8525
<edns>
```

In the display, <edns> is the **edns** prompt.

Task 4: Initialize the ns_helper database

Use the **edns init** command to initialize the **ns_helper** database that will reside on this node.

```
<edns> init
6 nodes responded to lcnode request
6 entries added to directory
0 names already existed 0 errors
```

Task 5: Verify that all nodes are in the database

Enter the **edns ls** command to list the database and verify that it includes all nodes on the network.

```
<edns> ls
grayling  brook       golden
gila      arctic_char         coho
6 entries listed.
```

Repeat Task 3 if some nodes were not added to the directory. Skip to Task 10 if you are only going to have one **ns_helper** node running on the network.

Task 6: Create an ns_helper replica

If you wish to run more than one **ns_helper** process, use the following command to set the default server process to a replica **ns_helper** node. In our example, we're setting the default server to the process running on **golden**, so that we can create an **ns_helper** database replica on that node.

```
<edns> set 8521
The default ns_helper is 0.8521
```

Task 7: Initialize the replica database

You must initialize the replica database (in the example, **golden**) with information from the original database (on **brook**).

```
<edns> init -from 8525
```

Task 8: Verify that the two databases are identical

Use the **edns diff** command to verify that the original and replica **ns_helper** databases contain the same information.

```
<edns> diff golden brook
The two directories are identical
The two replica lists are identical
```

If the databases are not consistent, repeat Task 7.

Task 9: Create additional replicas

Repeat Tasks 6 through 8 for each additional replica node that you are initializing.

Task 10: Quit the edns program

End the **edns** session by typing the following:

```
<edns> quit
%
```

PROCEDURE 2-10. *Adding Nodes to the ns_helper Master Root Directory*

Use this procedure when you add a node to the network. Also use this procedure to give a diskless node a name if the node was not on the network when the **ns_helper** database was initialized or updated. The procedure updates the **ns_helper** master root directory with the information for the new node and propagates the information to all replica databases. In the example, the nodes added to the master root directory are **laker, paiute, brown,** and **dolly_varden.**

Task 1: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Add nodes to the default ns_helper root directory

Use the following command to add each new node's name and ID to the default **ns_helper**'s root directory:

<edns> **add** *node_name node_id*

For example:

```
<edns> add laker 7206
<edns> add paiute 128c
<edns> add brown 3333
<edns> add dolly_varden 4b70
```

Task 3. List the root directory

Changes to the database take effect immediately. List the directory to be certain you did not make errors.

<edns> **ls -n**

```
nodeid name
7206   laker
3333   brown
4b70   dolly_varden
128c   paiute
503f   grayling
8525   brook
8521   golden
1c68   gila
  5f6   arctic_char
70de   coho

10 entries listed.
```

Use the **edns del** command to remove any errors you have made; then use the **add** command to correct information.

PROCEDURE 2–11. *Deleting Names from the Master Root Directory*

Use this procedure to remove a node from the network or to delete any entries that you added incorrectly to the master root directory. In the example, the nodes **laker** and **paiute** are deleted from the master root directory.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Delete entries

Use the following command to delete the entries that you want to remove:

```
<edns> del laker
<edns> del paiute
```

Task 3: List the root directory

Changes to the database take effect immediately. List the directory to be certain you did not make errors.

```
<edns> ls
brown  dolly_varden      grayling
brook  golden            gila
arctic_char              coho

8 entries listed.
```

PROCEDURE 2-12. *Changing a Node's Name*

Use this procedure if you change a node's name, for example if you change the name of node 3333 from "sunapee" to "speckled." Also use this procedure to give a diskless node a name if the node was on the network when the **ns_helper** database was initialized or updated. (In this case, the diskless node already has the default name **diskless_$***nnnnnn*, where *nnnnnn* is the node ID.)

Task 1: Uncatalog the old node name

Repeat this task at each node on the network. Otherwise, the node will be cataloged under both the new and the old name. Log in as user. Use the **uctnode** command to remove the node's old name from the root directory.

```
% uctnode sunapee -l
 "sunapee" uncataloged.
```

Task 2: If you are not logged on, log in at any node

Task 3: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 4: Delete the node's old entry from the master root directory

Use the **edns del** command to delete the node's old entry from the database.

```
<edns> del sunapee
<edns> del diskless_$009a7d
```

Task 5: Add new entries

Use the **add** command to create a new entry in the database with the node's new name.

```
<edns> add speckled 3333
<edns> add nodisk 9a7d
```

PROCEDURE 2–13. *Replacing Information for a Node in the Master Root Directory*

Use this procedure whenever you replace a node's disk with another disk and whenever you use the **invol** utility. These operations replace the node's ID, and you must put the new information in the database.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Replace the old information

Use the following command to replace the node ID information for the node named **coho**.

```
<edns> rep coho 70de
```

PROCEDURE 2–14. *Adding or Initializing an ns_helper Replica*

Use this procedure when you add an **ns_helper** replica. Use Tasks 2 through 5 to reinitialize the database of an existing **ns_helper** replica. In this procedure's examples, we initialize the new **ns_helper** replica on node **coho**, node ID 70de, from **brook**, node ID 8525.

Task 1: Start ns_helper on the new node

Use Procedure 2–8 to start **ns_helper** on **coho**.

Task 2: Invoke ns_helper and set the new default

At any node, enter the following command to invoke **edns** and set the default **ns_helper** directory to the new replica node, **coho**.

```
% edns 70de
The default ns_helper is 0.70de
```

Task 3: Initialize the replica database

Use the following **edns** command to initialize the replica database from the existing **ns_helper** replica at the node **brook**.

```
<edns> init -from 8525
```

Task 4: Verify master root directories

Use the following **edns** command to verify that the master root directory and replica list are the same on both the original replica node and the new replica node.

```
<edns> diff coho brook
The two directories are identical
The two replica lists are identical
```

If the databases are not consistent, repeat Task 3.

PROCEDURE 2–15. *Checking Replica Node Clocks*

You must keep the clocks on **ns_helper** nodes synchronized because the **ns_helper** applies time stamps to the information in its database. Skewed clocks can result in new data from an **ns_helper** node with a slow clock being deleted and replaced by older (and inaccurate) data from a replica node with a fast clock. We recommend that you keep replica node clocks within one minute. **The ns_helper** allows more than one minute of divergence and will indicate the skew after the range exceeds five minutes.

You should check the replica nodes' clocks daily until you know how long it takes for the clocks to diverge. You can then adjust the replica nodes' clock inspection schedule accordingly.

To check the **ns_helper** node clocks, perform the following tasks:

Task 1: Invoke edns

From any node, enter the **edns** command:

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: List replicas' clock times

Enter the following **edns** command:

```
<edns> lr -clocks
```

The following message from **lr** verifies that the replica nodes' clocks are synchronized well enough for **ns_helper** to operate properly:

```
 replica   datetime
 0.0070de  86/08/09.16:52
 0.008525  86/08/09.16:53
 0.008521  86/08/09.16:54
All clocks are synchronized to within ns_helper threshold.
```

The next message indicates that clocks are skewed:

```
 replica   datetime
 0.0070de  86/08/09.16:51 clock skewed
 0.008525  86/08/09.16:58 clock skew warning
 0.008521  86/08/09.17:03 clock skewed
```

Task 3: Synchronize skewed clocks

If the clocks are skewed, use Procedure 2–7 to synchronize them. Then use Procedure 2–16 to check the replica databases for inconsistencies and, if necessary, make them consistent.

PROCEDURE 2–16. *Maintaining Consistency of Replicated Databases*

The example used in this procedure compares and unifies the databases on the **golden** and **brook** nodes.

Task 1: Compare Replica Databases

Use the **edns diff** command to check whether two replica databases are the same. This command shows any discrepancies in names, UIDs, or addresses. For example, it reports any discrepancies that might have been caused by skewed clocks. It also reports any differences in the replica lists.

```
<edns> diff golden brook

                value in          value in
              0.008521          0.008525
diff              directory         directory              name

name            name found      name not found       gila
 uid   21089D87.4000503f   2108af41.4000503f       grayling

 The two replica lists are identical
```

In this example, **diff** shows that **gila** is cataloged in **golden**'s master root directory but not in **brook**'s. It also shows that **grayling** has a different UID value in the two replica master root directories.

Repeat the **diff** command until you have compared all replica nodes. For example, if **ns_helper** runs on **golden, brook,** and **coho**, the following two commands compare all replicas:

```
<edns> diff golden brook
<edns> diff golden coho
```

Task 2: Unify Replica Databases

The **edns** utility provides two commands for making replica databases consistent: **merge** and **merge_all.**

The **edns merge** command updates one replica database from a second database. It operates on both the directory and replica list of an **ns_helper**. In the following example, **merge** adds to **brook**'s database any information that is present in **golden**'s database but absent from **brook**'s. It also replaces any information in **brook**'s database that is older than information about the same entry in **golden**'s database. It is important to note that nothing is changed in the **–from** replica (for example, **golden**'s) database. Type

```
<edns> merge brook –from golden
```

Since **merge** operates on only the target replica database, it is often appropriate to merge all replicas into a consistent state. The **edns merge_all** command merges all replica databases, using either the default replica node or a specified node as the source node. The command merges the information from all **ns_helper** replicas on the source node's replica list into the source node's database, using the most

recent information whenever there are conflicts. It then updates all other replica databases with the contents of the source node's database. However, it can only get information from and update **ns_helpers** that are on its replica list. Therefore, check to make sure that the source node's replica list includes all replica nodes before using **merge_all**. The following steps use **merge_all** to ensure that all databases are consistent.

1. Invoke the **edns** utility.

   ```
   % edns
   The default ns_helper is 0.8521
   ```

2. Enter the **lr** command to list names of the nodes in the default **ns_helper**'s replica list.

   ```
   <edns> lr
   replica
   0.008525
   0.008521
   ```

3. If any **ns_helper** replica nodes are missing from the list, use the **addrep** command to add the node. For example, the replica list in Task 2 is missing **coho** (node ID 70de). Enter the following command to add the node:

   ```
   <edns> addrep coho
   ```

4. Merge all replica databases; enter

   ```
   <edns> merge_all
   ```

In this case, the default replica node (**golden**) is the source node, and this command merges the information from **brook** and **coho** into **golden**. It then merges **golden** into **brook** and **coho**.

PROCEDURE 2–17. *Removing an ns_helper Replica*

Use this procedure to stop an existing **ns_helper** replica process, delete the node's replica database, and delete the node's name from all other **ns_helper** nodes' replica lists.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Delete the replica

Use the **edns delrep** command to delete the replica.

```
<edns> delrep coho
```

The **delrep** command deletes the specified node's ID from the replica list of all other **ns_helpers**. It also causes the **ns_helper** at the specified node to delete its database and stop active service to the network. The inactive replica does not accept new transactions and will only accept **edns info** requests. It continues to run until it has completed sending any information it has that has not yet been propagated to the other replica nodes. When all information has been sent, it stops running.

Task 3: Check for running server

Use the **pst** command to see if the server process is still running from time to time. If you are at the replica node, enter

```
% pst
```

If you are at any other node, enter

```
% pst -n replica_node_specification
```

If **ns_helper** is not mentioned in the process list, it has stopped running.

If the deleted **ns_helper** is still running after a few days, you may have to stop it manually. This is the case if the deleted **ns_helper** has stopped, but the node is rebooted before you do Task 4; the stopped **ns_helper** restarts when the node reboots. Use the **edns info** command to see if the **ns_helper** can now be stopped.

```
% edns 70de
The default ns_helper is 0.70de
<edns> info
The default ns_helper is 0.70de
Its status is uninitialized.
```

If the **info** command reports that the deleted replica **ns_helper** is uninitialized, then it is appropriate to stop it.

```
<edns> shut 70de
```

Task 4: Disable ns_helper startup

When the process has stopped, remove the following line from the appropriate start-up file on node 70de (**coho**).

```
# cps /sys/ns/ns_helper
```

This prevents the **ns_helper** process from retstarting on **coho** the next time the node is rebooted.

PROCEDURE 2-18. *Shutting Down an ns_helper Replica*

Use this procedure to immediately shut down an **ns_helper** and delete its database, without deleting the replica node ID from other **ns_helper**'s replica lists. Any information that the **ns_helper** has not yet sent to other replicas will not be sent and the information may be lost. Therefore, you should use Procedure 2-16 before you delete this replica.

Task 1: Invoke edns

From any node, enter the **edns** command.

```
% edns
The default ns_helper is 0.8525
<edns>
```

Task 2: Shut down the replica

```
<edns> shut coho
```

2.4 Remote Process Creation: The Server Process Manager

The Server Process Manager, **spm** (/sys/spm/spm), allows you to create a process on a node from another, remote node. On a DSP, **spm** starts when the operating system is loaded, and so it runs whenever the DSP is online. **Spm** also starts the **mbx_helper** program. Since they have no monitors or keyboards, DSPs would be unusable without both of these server processes. Once **spm** is started, you can create processes from a remote node by using the shell command **crp**, as well as log in to the node for debugging purposes or to maintain servers.

2.4.1 Starting and Stopping spm

To start **spm** from the DM command line, enter the following:

```
Command:  cps /sys/spm/spm -n server_process_manager
```

The **spm** begins immediately and continues after logout.

To invoke it from the **/etc/rc** start-up file, uncomment the following line in the file:

```
# cps /sys/spm/spm -n server_process_manager
```

The **spm** begins when the node is booted, and it continues under normal conditions until it is intentionally stopped with the shell command, **sigp** as shown:

```
% sigp server_process_manager -q
```

2.4.2 The shutspm Command

In addition to the **sigp** command, you can also use the **shutspm** command to shut down the Server Process Manager on a remote server node. Unlike **sigp**, which shuts down only the **spm** process, **shutspm** shuts down the **spm** and then performs an orderly shutdown of the node. When you reboot the node after a **shutspm** shutdown, it reboots without performing a **salvol**. (If you require a **salvol**, shut down the node by pressing its RESET button.)

To shut down the **spm** with **shutspm**, create a remote process (via the **crp** command) on the target node and enter the **shutspm** command.

To prevent **spm** from responding to the **shutspm** command, add the following line to the **/etc/rc** file on the node or the DSP:

```
no_shutspm
```

The spmshut_ec File

The **shutspm** command performs the shtudown by advancing an eventcount file, **'node_data/spmshut_ec**, to the point that executes an orderly node shutdown.

The **spm** creates the **spmshut_ec** file in the **'node_data** directory. If the default ACL for files created in this directory is %.%.%, **spm** will apply the following protection to the **spmshut_ec** file:

Subject ID	Access Rights
%.sys_admin.%	pwrx
%.%.%	r

This ACL limits **shutspm** shutdown to sys_admin log-in accounts, but permits any account to delete the **spmshut_ec** file whenever **spm** is not using it. If the default ACL for **'node_data** has been changed, **spm** creates the eventcount file with that default ACL.

If the **spmshut_ec** file already exists when **spm** starts up, **spm** does not change its ACL. This ACL application procedure provides some control over who may shut down a remote server while still allowing you to administer your system the way you choose.

2.5 BSD Mail Services

BSD users can use Domain Professional Support Services (DPSS™) Mail as their mail delivery system. This subsection briefly describes DPSS/Mail™. For more detailed information on using and administering DPSS/Mail, see the *DPSS/Mail User's Guide*.

DPSS/Mail uses a subscriber directory to identify mail addresses. This directory is a field associated with each user's entry in the registry. You must ensure that the directory contains each mail user's mail address (using the **edsd** command).

DPSS/Mail can be used to send mail to UNIX systems, sendmail, and Alis* systems.

2.5.1 DPSS/Mail and UNIX Mail

A UNIX mail gateway, supplied with DPSS/Mail, can be accessed by sending mail to addresses of the form *user@unix* (for DPSS/Mail to UNIX systems) and *user@dpss* (for UNIX systems to DPSS/Mail). These addresses must be in the subscriber directory.

In addition, the sendmail configuration file must be changed to identify the mailer for the DPSS/Mail or UNIX system gateways.

*Alis is a trademark of Applix, Inc.

2.5.2 DPSS/Mail and sendmail

An optional feature allows all DPSS/Mail to be delivered through **sendmail**. This option can be configured on a system-wide basis by setting the **sm** option in **mail/$config/ver.17**. See Chapter 8 for more information on **sendmail**.

2.5.3 DPSS/Mail and Alis

If you are using the Alis gateway you must install **sendmail**, since mail from DPSS to Alis is delivered via **sendmail**.

There are two types of configurations for mail that enters the Alis mail system: the user and the gateway configurations.

Users communicate with the gateway that allows communication with Alis, using **sendmail_post**. This configuration requires a line, similar to the following, in **sendmail.cf** to target the gateway node:

```
Malis,      P=/usr/lib/mailer/sendmail_post, F=lFDhum, S=10,
R=20, A=sendmail_post $u -n //alis_gateway_node
```

In the gateway configuration, **sendmail_deliver** executes on the node running Alis, which serves as the gateway. The **sendmail_deliver** process waits for messages and then executes the local **sendmail**. The gateway executes the Alis import program.

The **sendmail.cf** on the gateway node for Alis mail looks similar to the one below:

```
Malis,      P=/usr/lib/mailer/alis_import, F=lFDhum, S=10,
R=20, A=alis_import $u
```

You must start **sendmail_deliver** via the appropriate start-up file on the gateway node. For example, the simplest way to start **sendmail_deliver**, is to include the following line in the **etc/rc** file:

```
cps /usr/lib/mailer/sendmail_deliver
```

Chapter 3

Administering Nodes in the Network

Contents

Chapter 3
Administering Nodes in the Network

This chapter describes how to administer nodes in the network environment. It includes information about these topics:

- The directory structure of the network and individual nodes
- Information on providing network services to node users
- Node activity when the node starts up (boot time) and when a user logs in, and how to use start–up and log–in files
- Steps to troubleshoot node software behavior
- Reference information on servers used by nodes
- Information on user log–in accounting

3.1 The Network Directory Structure

The operating system expects information about nodes to be organized in a hierarchy called the **naming tree**. The two naming tree components, directories and files, combine to form pathnames. Directory names and file names must have fewer than 255 characters; pathnames must have fewer than 1023 characters. Figure 3–1 illustrates the network naming tree, including some of the standard software directories.

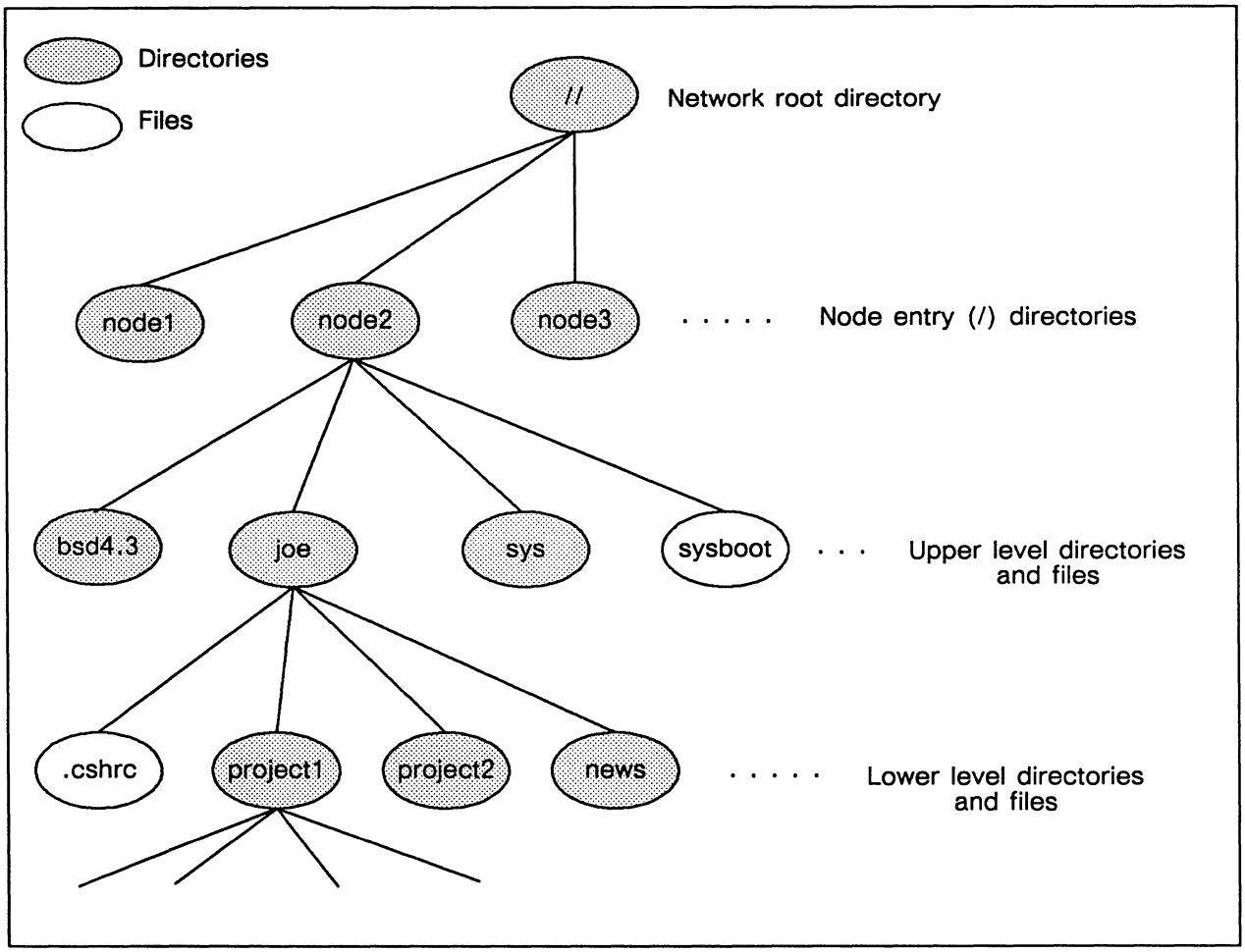


Figure 3–1. Network Directory Structure — the Naming Tree

3.2 Node Directory Structure

A node's directory structure comprises entry and root directories, upper–level directories, and volume entry directories. This section discusses these directory types, distinguishes between physical and logical volumes, and explains two special characters in the file system: the ' (tick) in **'node_data** and the / (slash) identifier in the context of the directory structure.

3.2.1 Node Entry Directories and Root Directories

The node entry directory is the highest–level directory in a node's naming tree that can contain files, links, and other directories. The entry directory name is the same as the disked node's name.

The set of node entry directory names in your network is the network root directory, sometimes referred to as the top-level directory. Every disked node has its own copy of the root directory that the operating system uses to find objects stored on other nodes.

3.2.2 Upper-Level Directories

Upper-level directories are one level below the node entry directory in the BSD naming tree structure. Upper-level directories contain BSD system software, system and programming libraries, and users' home directories. The system software installation procedures create the upper-level directories that the system needs to operate. In addition, you can use the **mkdir** command to establish upper-level directories on each node, such as home directories for different users.

You should restrict access to upper-level directories that contain system software by setting permissions on them so that they cannot be accidentally deleted or altered by users. Individual users may also want to protect all or part of their home directories by using permissions. Chapter 5 discusses protection.

3.2.3 Disk Volumes and Volume Entry Directories

Disked nodes have one or more **physical volumes**, that is, physical media. One disk drive is treated as a single physical volume; therefore, a node can have as many physical volumes as it has disk drive units. Each physical volume can be divided into **logical volumes**, independent partitions of the physical volume. You use the **invol** utility to create and maintain logical volumes on the physical media.

Most nodes have one logical volume per physical volume, because this is usually the most efficient way to use disk space. However, you might want to partition a physical volume into two or more logical volumes if you want to reserve part of a large disk for temporary files, or if you want to have another version of the operating system on the disk, or if you want to be able to run **salvol** on only part of the disk (so that **salvol** will run more quickly).

Logical Volumes on BSD Systems

BSD uses block special files (in the **/dev** directory) to define the logical volumes. The BSD installation procedure automatically creates three sets of block special files, one each for a single Winchester, floppy, and storage module logical volume.

You must use the **/etc/mknod** command to create additional block special files for new logical volumes. For example, if you install a second storage module drive on a file server node and use **invol** to partition the new disk into two logical volumes, you should create the following block special files:

/dev/sm1a	Block special file for logical volume 1 (a) of the second storage module unit
/dev/sm1b	Block special file for logical volume 2 (b) of the second storage module unit
/dev/rsm1a	Raw block special file for logical volume 1 (a) of the second storage module unit
/dev/rsm1b	Raw block special file for logical volume 2 (b) of the second storage module unit

You use the block special files with the **mount** and **umount** commands to mount and unmount the volumes. A volume must be mounted to be accessible. Any volume that you mount by using the **mtvol** command is also accessible to BSD, even if the volume does not have a descriptor file. The volume will also be listed in the **/etc/mtab** mounted volume table. Similarly, whenever you reset a disked node, the node ROM software automatically mounts the node's **boot volume**, that is, the logical volume that contains the system bootstrap software. As a result, the **/etc/mtab** table includes the boot volume.

Each logical volume that is mounted on a node has a **volume entry directory**. This directory is the logical volume's highest–level directory. The boot volume's volume entry directory is also the node entry directory. A volume's entry directory must be mounted on the next higher directory in the naming structure. The boot volume (node) entry directory is mounted on the root directory, the highest–level directory in the network naming structure. The **mount** command automatically mounts a volume's entry directory on its next higher directory when you mount the volume.

Because the name of a directory is mounted on the next higher–level directory, you can change the name of a volume entry directory each time you mount the volume. This is particularly useful when you are mounting floppy disks, because you might want to put floppy disks with different software at different locations in the naming tree. Thus, you might mount a floppy disk with data from an analysis of the performance of widgeta as **/tests/data/widgeta**. When you are done using this disk, you could unmount it and mount a disk with financial analysis results as **/finances/q286**.

Mounting a Volume on a Diskless Node

You can mount a logical volume on a node that you boot diskless—if the volume is physically located at the node. This capability is useful if a node's copy of the system software is corrupted and the node will not boot normally, but you must access data files on the node's local disk. To mount a disk on a diskless node:

1. Boot the node as a diskless node.
2. Use the **mknod** command to make a block special file in the partner node's **/sys/node_data.***diskless_node_id***/dev** directory for each logical volume to be mounted (if the files do not already exist).

3. Use the **mkdir**(1) command to create the volume entry directory if it doesn't already exist.

4. Use the **mount** command to mount the required logical volumes.

You must perform Step 2 because a diskless node's **'node_data** directory does not automatically have BSD block special files. For more information on the requirements for diskless nodes, see Section 3.11, "Administering Diskless Nodes," later in this chapter.

For example, assume node **apple** has a single Winchester disk with a single logical volume. If you boot **apple** as a diskless node with the partner **orchard**, you can use the following commands to mount **apple**'s disk in the root directory:

```
% /etc/mknod  /dev/wn1a  b  0  1

% /bin/mkdir  //apple

% /etc/mount /dev/wn1a //apple
```

It can be useful to mount a diskless node's boot volume in the root directory as we have done in this example. This way, you and others can refer to the volume's files by using the usual absolute pathnames, such as **//apple/helena/com/test.pas**. However, mounting a diskless node's boot volume in the root directory has no effect on the meaning of / (see the next section). This character still refers to the node entry directory of the partner node, **orchard**, and not to the **//apple** entry directory.

3.2.4 The 'node_data and / Identifiers

Using Your BSD Environment describes directories and links in detail. However, because the meaning of **'node_data** (tick–node_data) and / (slash) are especially important in the context of system administration, we go over the information here as well.

The meanings of the identifiers **'node_data** and / (alone or at the beginning of a pathname) are variable, and depend on the context in which you use them. Using them incorrectly, especially in links, can result in circular or otherwise incorrect references.

As the first character of a pathname, the slash character (/) always refers to the root directory of the node where the process is executing. Similarly, **'node_data** always refers to the **/sys/node_data[.***node_id***]** directory for the node where the process is executing.

The slash, /, and **'node_data** conventions are powerful tools. The **'node_data** convention allows you to specify the **node_data** directory of the node you are working on, and that node only, even if the node is diskless. Similarly, the slash character refers to your working node's entry directory, independent of the current working directory.

Note that in UNIX shells, the tic, ' ,is a special character and must, therefore, be preceded with a backslash (\) . This means, for example, that you must enter the following command to change your working directory to **'node_data/etc** in a UNIX shell:

```
% cd \'node_data/etc
```

Some examples of using / and **'node_data** follow.

The 'node_data Directory and Diskless Nodes

The actual pathname of the **'node_data** directory for a diskless node is distinguished from its disked partner's **'node_data** with a suffix that consists of its hexadecimal node ID. When you're working on a diskless node and specify **'node_data**, the operating system understands that to mean the **/sys/node_data.***node_id* directory on the disked partner.

Suppose there are three nodes: **brook**, **blue**, and **rainbow**, as shown in Figure 3–2. **Brook** is the disked partner node for the two diskless nodes **blue** and **rainbow**. A program or person working at node **brook** who reads the file **'node_data/startup.19l** will read the file **//brook/sys/node_data/startup.19l.** A program or person working at node **rainbow** (node ID e467) who reads the file **'node_data/startup.19l** will read the file **//brook/sys/node_data.e467/startup.19l.** A program or person working at node **blue** (node ID 632b) who reads the file **'node_data/startup.19l** will read the file **//brook/sys/node_data.632b/startup.19l.** (Remember, however, that a program or person working at either node **rainbow** or node **blue** who reads file **/sys/node_data/startup.191** will read that file on the **brook** node.)

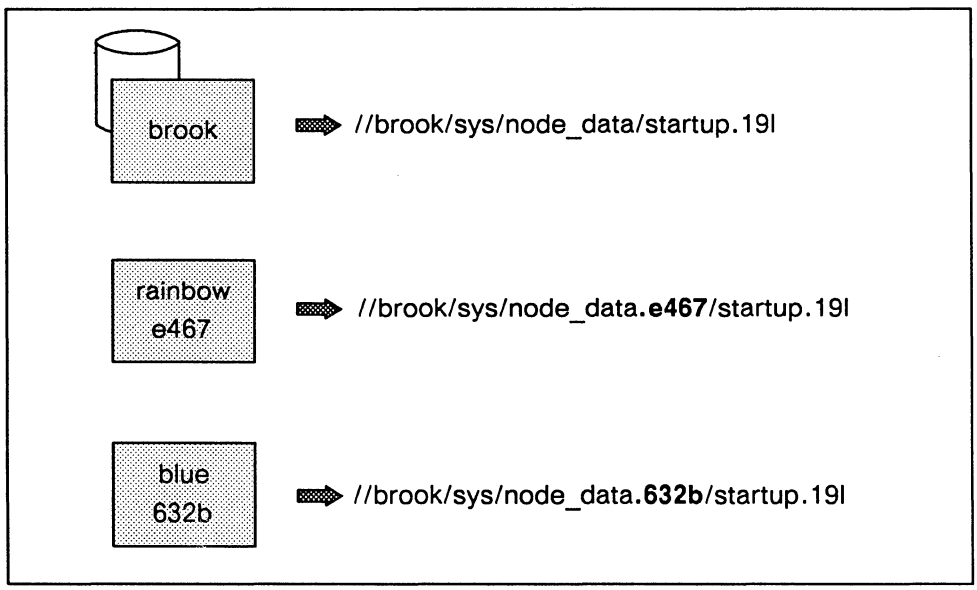


Figure 3–2. Reading 'node_data from Diskless Nodes

The crp, rlogin(1), rexec(3x), rsh(1), and rcmd(3x) Commands

If you use **crp**, **rlogin**(1), **rexec**(3x), **rsh**(1), and **rcmd**(3x) to execute commands or programs remotely, the commands execute at the remote node. Therefore, **'node_data** and **/** refer to the **node_data** and entry directories at the remote node. For example, if you are working at the node **here_node** and use **rlogin** to log in remotely on the node **there_node**, the following command displays the directory listing for **//there_node/sys**, the remote node's **/sys** directory.

```
% ls /sys
```

Circular and Unexpected References

Because **'node_data** and / have meanings that depend upon the location of the process that makes the reference, it is possible to use pathnames that result in circular references. This is particularly true when you use links to either the / or **'node_data** directory. For example, each node's **/tmp** directory is a link to **'node_data/tmp**. If you are working at node **bar**, you might try to use the following command to list node foo's **/tmp** directory.

```
% ls //foo/tmp
```

However, this command will actually list the contents of **//bar/sys/node_data/tmp** because **//foo/tmp** is a link to **'node_data/tmp**, and **'node_data** always refers to the **node_data** directory of the node executing the command, in this case node **bar**. Therefore, to list the contents of the **/tmp** directory of node foo when you are working at node **bar**, you must use the absolute pathname of the file in the command.

```
% ls //foo/sys/node_data/tmp
```

3.2.5 The Variant Link

Each node's **/bin**, **/usr**, and **/etc** directories are links to *(*$*systype)*/**bin**, (*$*systype*)/**usr**, and (*$*systype*)/**etc**, where *$systype* is the value of the system type environment variable. Therefore, if a process' systype is **bsd4.3**, the process executes commands in the **/bsd4.3/bin** and **/bsd4.3/usr** directories.

This use of variant links allows different processes to use different versions of Domain/OS simultaneously, and ensures that each process uses the correct versions of each command and call.

Symbolic Links

The BSD **ln**(1) command and the Aegis **crl** command create symbolic links that are identical objects and that are treated in the same way by BSD and Aegis commands. For example, you can delete a link created with either command with the **rm**(1) or the **dll** command. Both **ln** and **crl** create a link entry in the directory; the **ln -s** command does not create a new file that contains the linkage information.

3.3 Node Software Structure

Figure 3–3 shows a typical node's software structure, including system software and user entry directories, starting at the node entry directory level. This figure shows only the general way the software is organized; it does not include all software.

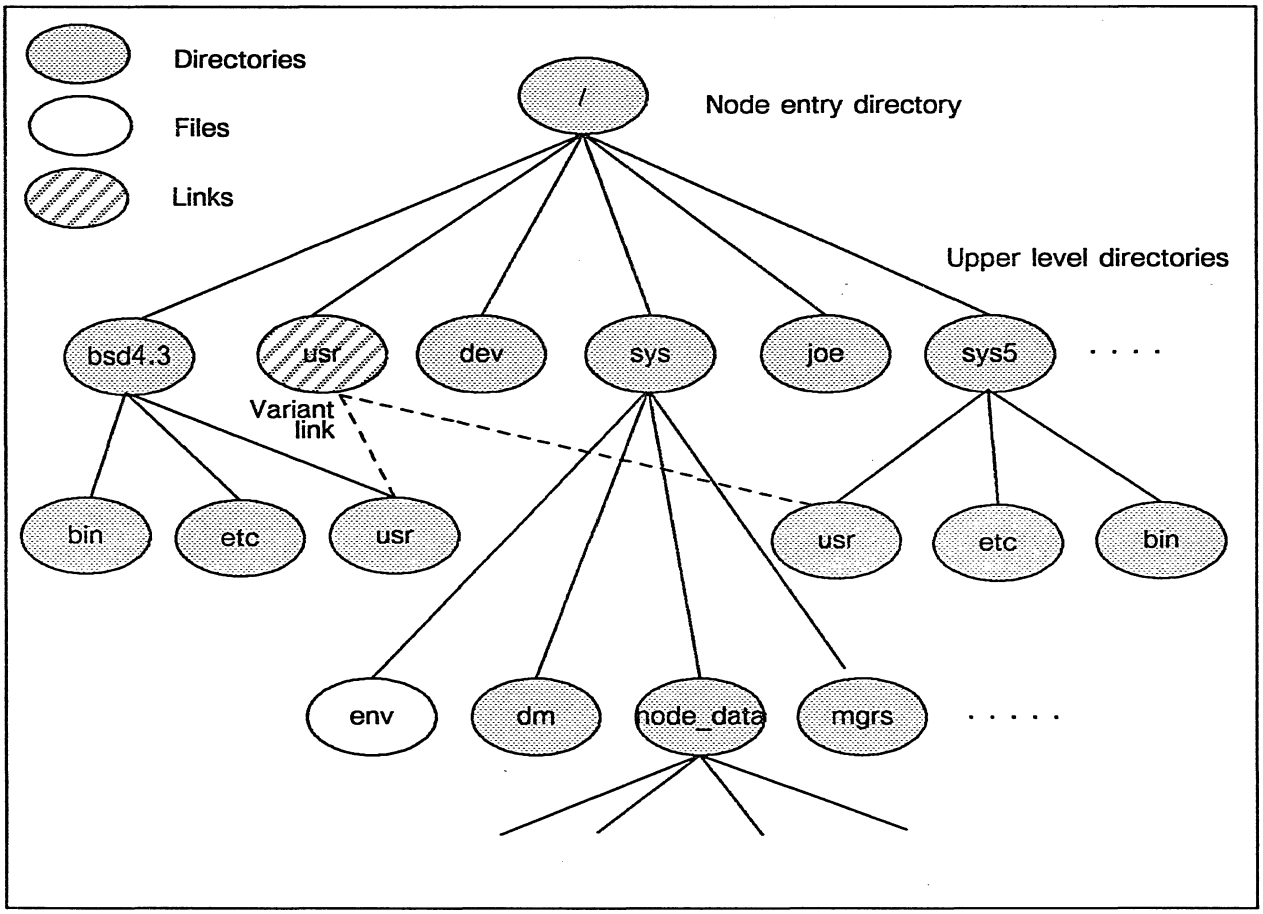


Figure 3–3. Disked Node Directory Structure

The exact structure of your directories and their contents will differ depending on the exact configuration of your system.

In the figure, the node entry directory contains the BSD system software and other subdirectories that themselves contain BSD system software. The /sys directory contains system software and many of the directories used for node and network management. Two subdirectories of the /sys directory, the Display Manager directory **/sys/dm** and the network management directory **/sys/net**, contain files and programs you use to create and administer network services. All installations will have the **'node_data** directory.

3.3.1 The 'node_data Directory

Every node, disked or diskless, has a **'node_data** directory. This directory contains files that the node needs to perform many system functions. These files include start–up and configuration files, per–node system files, interprocess communications mailboxes, and device files.

The actual pathname of a disked node's **'node_data** directory is

//nodename/sys/**node_data**

The pathname of a diskless node's **'node_data** directory is

//partner_node/**sys/node_data.***node_id*

The optional *.node_id* part of the file name enables a disked node to be a partner to one or more diskless nodes.

For example, if the disked node **golden** is the partner node for diskless nodes **sunapee** (node ID efd3) and **char** (node ID f2a4), **golden** would have the following three directories, each containing system operation information for its respective node:

/sys/node_data	(for **golden**)
/sys/node_data.efd3	(for **sunapee**)
/sys/node_data.f2a4	(for **char**)

NOTE: In later examples, when we refer to a node's **/sys/node_data** directory, we use the syntax **/sys/node_data[.***node_id***]** when we want to show that we are talking about the **/sys/node_data** directory for both disked and diskless nodes.

See the "Administering Diskless Nodes" section later in this chapter for further information about managing the **/sys/node_data.***[node_id]* directory for diskless nodes.

3.3.2 Directories for Temporary Files and Log Files

Apollo provides two directories for temporary files, **'node_data/systmp** and **'node_data/tmp**, and one for log files, **'node_data/system_logs**.

The **'node_data/systmp** directory is used for temporary operational files created by Apollo software (system and other). The **'node_data/tmp** directory (a link from **/tmp**) directory is used for temporary files created by the UNIX operating system. Files in both of these directories are deleted by the **/etc/rc** file when the system starts up.

The **'node_data/system_logs** directory is used by Apollo software (system and other) to store log files.

You should ensure that the files in all of these directories are writable by all users. In addition, you should ensure that the **'node_data/system_logs** directory is purged occasionally of uneeded files so that it stays a reasonable size.

By separating temporary directories from the other directories and files in **'node_data**, you can set protection for temporary files so that users can access them, but still maintain the overall tighter protection established for the **'node_data** directory.

NOTE: All systems, including nodes that do not have a SysV environment installed, have a **/sys5.3/bin** directory. On systems that have only BSD or Aegis, this directory contains a minimum subset of SysV commands that are required to install software on all nodes, independent of the installed environments. This **/sys5.3/bin** directory enables solution suppliers to provide a single (SysV) script that will correctly install software on all nodes, independent of the installed environment.

3.4 The BSD Environment

This section describes elements of the BSD environment that are especially important to system administration.

3.4.1 The DM and Context Inheritance

Whenever you execute a Display Manager command, the DM inherits its context from the last active display window. This context includes all environment variables, including the SYSTYPE value. It also includes the current working directory. This context inheritance has several importance effects, including the following:

- The DM does not necessarily inherit the context from the window that currently has the cursor, if you have just moved the cursor into the window. You must initiate some activity in the window (if only by pressing the space bar when the cursor is in an input pad) before the DM can recognize the window as "current."

- The DM inherits the context from the window that most recently sent it input. Therefore, if you start some command (say an **ls –l** of a long directory) in a BSD C shell, move to a SysV Bourne shell and execute another command (say, the **pwd** command) and then move to the DM input window, a DM **env SYSTYPE** command will return the value SysV, even if the C shell has not completed the directory listing.

- Any process that you create by executing a DM **cp**, **cpo**, or **cps** command inherits its environment variables from the DM, and therefore from the current DM context.

- Because the DM context includes the working directory, the meaning of a relative pathname (for example, in a **cv** or **ce** command) depends upon the working directory of the most recently active window.

3.4.2 Special Administrative Considerations

The following sections describe special administrative concerns for the **sendmail**(8), **syslog**(8), and **tip**(1) programs. For additional information about **sendmail**, see Chapter 8. For information on **tip**, see the *BSD Command Reference*.

The sendmail and syslog Programs

The **sendmail** process uses TCP/IP to send status messages to **syslog**. If a **syslog** daemon does not run on the same node as **sendmail**, then the messages are lost. If **syslog** is running but **sendmail** cannot open a socket (for example, if **tcpd** is not running on the **sendmail** node), then **sendmail** writes the message to **'node_data/syslog**.

The tip Program

The following considerations apply to **tip**:

- You can use **tip** only on a node that is physically connected to a modem.
- Each **tip** node must have a separate **/usr/admin/aculog file**. At many installations, BSD is installed so that each node's **/usr/admin** directory is a link to **/usr/admin** on an administrative node. In this case, you must make **/usr/admin/aculog** on the administrative node a link to **'node_data/aculog** on each **tip** node.
- You must make sure that the **/etc/remote** file at each node that uses **tip** is correct for the configuration at that node. If you have multiple **tip** nodes, you must either create multiple **/etc/remote** files or standardize your file and modem installations.

  If you have multiple files, **/etc/remote** on the administrative node must be a link to **'node_data/etc/remote**. Each **tip** node must then have a **sys/node_data***[.node_id]* **/etc/remote** file.

  If you use a single **/etc/remote** file, you must standardize the use of SIO lines and the contents of the **/etc/remote** file so that there are no configuration conflicts.

3.5 Using Nodes to Distribute System Resources

When you administer a Domain network, you manage system–wide file resources, such as databases and libraries, and manage processes that provide network–wide services (server processes).

3.5.1 Managing System Resources

You must manage system resources in order to distribute network resources such as databases, organize and protect user home directories, and organize and protect libraries of application software.

Frequently you manage system resources by creating and managing upper–level directories. The decisions you make about the locations of upper–level directories that contain network resources will affect the performance of your network. For example, some system libraries and databases are large or heavily used; you should position such resources strategically to maintain an even flow of network traffic and optimize disk space.

Using Upper–Level Directories

By assigning each library of application software its own upper–level directory, you can easily protect the software from accidental or unauthorized changes. You can set the permissions for the libraries so that only system administrators have full rights to change contents. For users who only run application programs, you can set their **home directories** to the proper application program library.

A **home directory** is an upper–level default working and naming directory set by the operating system when the user logs in. Use the registry entries, as described in Chapter 4, to specify a user's upper–level directory as a home directory. Refer to *Using Your BSD Environment* for a more detailed discussion of home directories.

Creating Upper–Level Directories

You create upper–level directories in the same manner as you create other directories, by entering the **mkdir**(1) command and specifying the directory pathname. For example, to create the upper–level directory **joe** on the node **//joes_node**, enter:

```
% mkdir //joes_node/joe
```

You can then use **chown**(8), **chgrp**(1), and **chmod**(1) to set the ownership and permissions as required for the directory. (Remember that you must be **root** to execute **chown**).

3.5.2 Providing System Services

Server processes provide services to some or all of the nodes on a network. Servers normally run regardless of log–in and log–out activity. Server processes manage requests from **clients**, which may be programs or other processes. Clients request access to network resources such as data, peripheral devices, or communication pathways outside the network.

Server Processes and DSPs

Server processes often run on Domain Server Processors (DSPs), server nodes that do not have displays and that are dedicated to running these processes. However, you can configure network server processes on any kind of node.

Numbers and Locations of Servers

The number of times you implement a server process depends on the particular requirements of your site. You should run server processes that manage network databases, **ns_helper** for example, on several nodes to ensure user access. For example, an **ns_helper** process must run on each Domain network that is connected to make a Domain internet. You might also want to run **ns_helper** on network loops that are frequently separated from a main network.

Decisions about the number of times to implement a server are closely related to the placement of server nodes within the network topology. For example, five printers can be managed from one, two, or five server nodes, depending on the locations of the printers. If your network covers a one–story building and a larger two–story building, you might want three server node locations, one in the small building and two in the larger building.

The location of servers affects your ability to provide services to nodes and loops as they are switched in and out of the network. Note that a node selected to run a network server process can be used for other activities. It is not necessary, and usually not desirable, to place all network services on one or two nodes. Servers should run on nodes that are stable and secure. (You would not normally run **ns_helper** on a node in an open computing

room or a system development node.) Become familiar with the principles of network management and troubleshooting before you determine the numbers and locations of servers.

Later sections in this chapter discuss special considerations for BSD servers and nodes that provide other network–wide services. Other sections discuss the node start–up and log–in processes, and indicate methods for starting server processes at these times.

3.5.3 Server Process Information

Server processes manage requests for data access and data transfer, gather network statistics, manage access to network resources such as printers, and manage communication paths outside the network.

At the end of this chapter you'll find reference material on node server processes. Each server's reference pages contain the following information:

- A description of the server process
- Methods for starting, stopping, and reinitializing the server process
- Information about configuration files
- Information about options and arguments, and examples of their use
- Special considerations

Two servers, **ns_helper** and **netmain_srvr**, have interactive tools. The **edns** utility, which the system administrator uses with **ns_helper**, is described in the *BSD Command Reference*. The **netmain** tool, which is used with **netmain_srvr**, is described in Appendix A.

The reference information differentiates between creating servers on nodes with displays and those without displays (DSPs) because the two types of nodes sometimes require different start–up methods.

3.5.4 Methods of Starting Servers

Several methods are used for creating servers. The method used affects the server's attributes, whether it runs in the foreground or background, and whether it runs on a local or a remote node.

Starting Servers on a Local Node

You can start servers on a local node in the following ways:

- Inserting the server's pathname in the **/etc/rc**, **/etc/rc.user**, or **/etc/rc.local** files. These files are executed as a part of start-up processing. This is the recommended way to start server processes. See Section 3.8, "Start-Up Procedures," later in this section for more information.

- Executing DM create process commands from the from the DM input window. The DM **cp** (create process in a window), **cpo** (create process only), and **cps** (create process server) commands start server processes on a user's node. Commands issued from the DM input window start server processes immediately.

- Executing the **/etc/server** command. If the DM is not available, you can use this command to start server processes. This command has the following syntax:

  /etc/server *server_name server_arguments* **&**

  If you use the **&** option, **/etc/server** starts servers with exactly the same attributes as servers started with the DM **cps** command. If you do not use **&**, it starts them as background processes. Note also that **/etc/server** has a **-p** option that allows you to start servers with the log-in SID rather than **user.server.none**.

Note that you can also start servers by inserting the appropriate DM create process commands in the **'node_data/startup**[*.type*] (for DM) and **'node_data/startup.spm** (for SPM) file. This method, however, is not recommended, since future versions of BSD may not allow server startup in **'node_data/startup** files.

Refer to the *Domain Display Manager Command Reference* for complete information about the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**.

Starting Servers on a Remote Node

If the Server Process Manager (**spm**) is running on a node, you can create processes on that node from another location. (The **spm** runs on DSPs by default, so you can always create other servers or processes on a DSP from a remote node.)

To create processes from a remote node, use the shell command **crp** (create remote process). The **crp** command can take the **cp**, **cpo**, and **cps** local process commands as options to specify the attributes of the process created. For example the command

```
% crp -on //trout -me 'cps /etc/ncs/glbd'
```

starts the process named **glbd** (in the **/etc/ncs** directory) on the remote node named **trout**.

Refer to the *Domain Display Manager Command Reference* for complete information about the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**.

Summary of Server Process Start-Up Methods

Table 3-1 summarizes information about the ways to start servers.

Table 3–1. Server Process Start–Up Methods

Server Start–Up Method	**Process runs in foreground or background?**	**SID of Process** (*id* = node ID)	**Process runs on local or remote node?**
Paths to DM command files inserted in a start–up file:			
cpo	Background, runs whether or not anyone is logged in.	log–in account SID	Local
cps	Background, runs whether or not anyone is logged in.	log–in account SID	Local
DM commands, entered in the input window:			
cp	Foreground, ends on logout.	log–in account SID	Local
cpo	Background, ends on logout.	log–in account SID	Local
cps	Background, runs after logout (except for the **siologin** server).	**user.server.none.id**	Local
Shell commands, if spm is running on the remote node:			
crp –cpo	Background, ends on logout.	log–in account SID	Remote
crp –cps	Background, runs after logout.	**user–server.none.id**	Remote
etc/server command (can be used in the event the DM is not available)	Background, if the **&** options is used. Runs after logout (except for the **siologin** server). Foreground if the **&** option is not used.	**user.server.none.id** If the **–p** option is used, the SID will be the log–in SID.	Local

3.5.5 Naming Server Processes

When you use the **-n** *server_name* option with the server creation commands, you can provide a name for the server process. If you give a server process a name, you'll be able to easily identify its server in the list displayed by the **ps** (process status) shell command. If you do not give the server a name, and it does not name itself by default, the operating system identifies it with a process number. We recommend that you use names for servers (either the defaults that some servers provide, or a name of your choice).

For example, the following **cps** command creates a shell process that takes a server name as an argument:

% **cps /com/sh -c** *'server_name'*

With this command, although the shell process controls the server, the server has the attributes of any process started with the DM command **cps**. And as with any shell process, a server process created with this syntax can use command-line features.

3.5.6 Using Shell Command-Line Features

While most server processes that start in the DM command line don't use shell command-line features, there are some that do. For such servers, the DM passes command-line standard options to the server program. See the individual server descriptions for information about servers that use command-line features from the DM command line.

3.5.7 Maintaining Existing Servers

To check on the status of network server processes, use the shell command **pst**. The **pst** command lists all processes running on a node. The **-n** *node* option for **pst** shows the processes running on a remote node. Use **pst** and its options if you suspect that a server is not running.

If the **pst** display does not show the server process, restart the server.

If **pst** does not list a server that you started, the server might not have started properly. ACLs that don't allow access to the server process SID or to the **'node_data** directory can prevent a server from starting. If you use the ACLs that we provide in the ACL templates, you should not experience this problem. However, if you change ACL entries and then a problem with server startup occurs, check the ACLs assigned to the server program itself (for example, **/sys/alarm/alarm_server**) and the **'node_data** directory of the node on which you want to start the server. Any person (or process) starting the server program must have Execute rights to the server program, and Read and Write rights to the **'node_data** directory.

The **pst** command sometimes lists a server process as running even though the server has stopped. If this happens, stop the server process explicitly with the **sigp** (signal process) command, then restart the process from the DM command line. To stop a server process running on a remote node, use the **crp** command to log in to the remote node, and then use **sigp** to stop the process. Refer to the *BSD Command Reference* for information about the **sigp** command.

3.6 Establishing a Node's Environment

During software installation, the system administrator selects the Domain/OS environment(s) (that is, Aegis, BSD, or SysV) that will be available on the node. When Domain/OS software is installed on a node, you must determine the node's primary environment and system type.

3.6.1 The Node's Primary Environment

When multiple environments are installed on a node, one must be designated as the node's default primary environment. A value called "ENVIRONMENT" is used to specify the environment for all behavior except UNIX name resolution. ENVIRONMENT determines:

- Default key definitions
- User's default shell if no shell is specified in the user's registry entry
- Default path definitions

The ENVIRONMENT value can be **aegis**, **bsd**, or **sys5**. (Note that ENVIRONMENT is not an environment variable, but is maintained as a per-process value.) If only one environment is installed, that automatically determines the value of ENVIRONMENT. No choice is needed and changing the value is not meaningful.

3.6.2 The Node's SYSTYPE

SYSTYPE is an environment variable used to specify UNIX name resolution for many variant links that use $(systype). Valid values for the environment variable are **bsd4.3**, **sys5.3**, or blank.

3.6.3 The /etc/environ File

ENVIRONMENT and SYSTYPE values are specified in **/etc/environ** (which is actually a link to '**node_data/etc/environ**). For example, the file might say:

```
ENVIRONMENT = aegis
SYSTYPE = sys5.3
```

Valid ENVIRONMENT and SYSTYPE values are governed by what is installed on the node and the selection of the primary environment at installation time. Table 3-2 shows the valid combinations of primary environment, additional environments, ENVIRONMENT, and SYSTYPE.

Table 3-2. Relationships of ENVIRONMENT and SYSTYPE Values

Primary Environment	Additional Environment(s)	Value of ENVIRONMENT	Value of SYSTYPE
Aegis	None	aegis	blank
Aegis	BSD	aegis	bsd4.3
Aegis	SysV	aegis	sys5.3
Aegis	BSD and SysV	Aegis	Choice of bsd4.3 or sys5.3
BSD	NA	bsd	bsd4.3
SysV	NA	sys5	sys5.3

NOTE: If ENVIRONMENT is set to one of the UNIX values, SYSTYPE must correspond to that value.

If the **/etc/environ** file is not found, the defaults are ENVIRONMENT = **sys5** and SYSTYPE = **sys5.3**. The value of ENVIRONMENT can be displayed by using the **/etc/environment** command; thus, shell specific scripts can query the current environment.

3.7 Establishing a User's Environment

If a node has multiple environments installed, users can choose to change the primary environment in effect while they are logged in at that node. The file **$(HOME)/.environ**, like the **/etc/environ** file, can contain lines specifying ENVIRONMENT and SYSTYPE. If this file is found during user log-in processing, the environment and system type specified are used as the user's environment, as long as the specified environment is installed on the node. (If the environment is not installed, the node default environment is used.)

3.8 Start-Up Procedures

This section describes node start-up procedures, user log-in procedures, and related files.

3.8.1 Node Startup

The **/etc/init** program initiates node startup. This program first executes a Bourne shell script named **/etc/rc** (which, in turn, will execute other scripts) and then executes **/etc/ttys** (which also executes other scripts). Figure 3–4 illustrates node start–up procedures. Each script file in the start–up process is described following the figure.

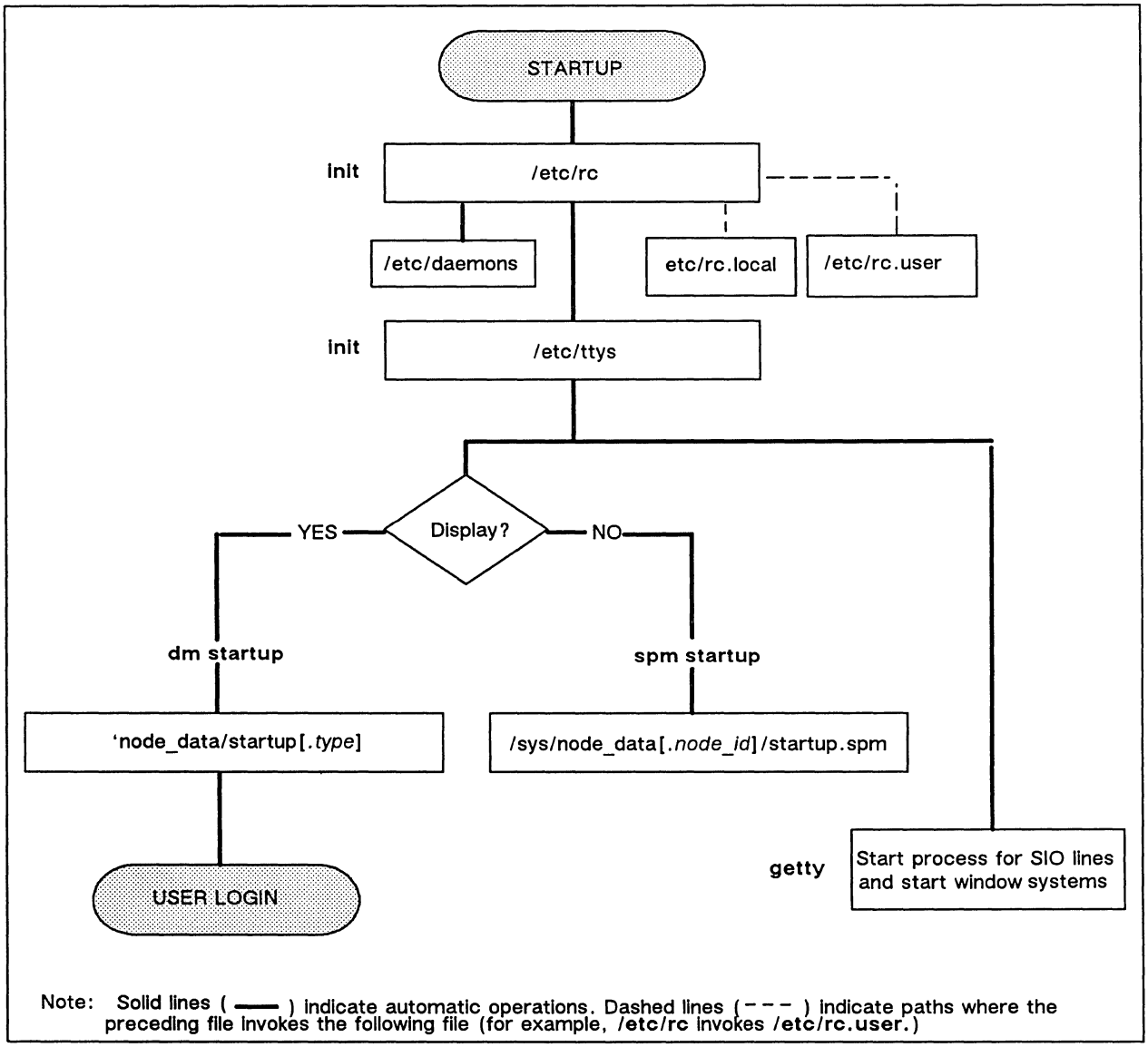


Figure 3–4. Node Start–Up Files and Operations

3.8.2 The /etc/rc File

The **/etc/rc** file is a Bourne shell script executed as the first step in node startup by **/etc/init.** The **/etc/rc** file starts servers and other global processes as **root.wheel.none.** This file must also contain the commands that

- Delete any files in the **'node-data/tmp** and **'node_data/systmp** directories
- Start any servers that require root access rights
- Invoke the **/etc/rc.local** and **/etc/rc.user** files (described in the next subsections)

Starting Servers in /etc/rc

The **/etc/rc** file contains lines that invoke servers. When the file processes these invocations, it first looks in the **/etc/daemons** directory to see if a file named for the server exists there. Since **/etc/rc** is owned by root and you must be root to edit it, **/etc/daemons** lets ordinary users (without root access) control which processes run from **/etc/rc** on their nodes.

The **/etc/daemons** directory contains stub files named for the servers that should be started. Unless a file with the server's name is in this directory, the server will not be started, regardless of whether it's in the **/etc/rc** file. Node users can add or delete files to control the starting of server processes by **/etc/rc.**

3.8.3 The /etc/rc.local File

The **/etc/rc.local** file, invoked by **/etc/rc** during start-up processing, starts TCP/IP and its related servers. The commands in **/etc/rc.local** will execute with root access rights. Note that the **/etc/daemons** directory must contain a file named **tcpd** for the TCP/IP server to start. See *Using TCP/IP Network Applications* and *Configuring and Managing TCP/IP* for more information on TCP/IP.

3.8.4 The /etc/rc.user File

The **/etc/rc.user** file, invoked by **/etc/rc** during start-up processing, contains commands to start servers and other global processes that run as **user.server.none**, not as root. This file can edited by users with other than root access, and does not need entries in **/etc/daemons** for the servers it starts.

3.8.5 The /etc/ttys File

The **/etc/ttys** file is executed by **init** during start-up processing. The **/etc/ttys** file starts the DM or SPM, getty processes for specified SIO lines, and possibly window systems.

Figure 3–5 shows the contents of a sample **/etc/ttys** file.

```
console   "/etc/dm_or_spm"        apollo     on         secure
tty01     "/etc/getty d1200"      dumb       off        secure
tty02     "/etc/getty std.9600"   dumb       off        secure
tty03     "/etc/getty d1200       dumb       off        secure
```

Figure 3–5. Sample /etc/ttys File

The first entry starts the DM or the SPM, depending on whether the node has a display. The remaining lines in the file enable **getty** services on SIO lines. DM and SPM startup, SIO line initialization, and window system startup are described in the following subsections.

3.8.6 Display Manager Startup

When the DM manager starts, it automatically executes '**node_data/startup**[*.type*]. The *.type* option identifies the display type. Although you can use this file to start server processes, we recommend you use **/etc/rc.user** and **/etc/rc.local** instead. (If you do start processes in this file, the processes run regardless of log–in and log–out activity and run as **user.server.none**.)

Figure 3–6 shows the first lines in the start–up file we provide for the DN3000 monochrome node; the file is named '**node_data/startup.1280bw**. Corresponding start–up files for other types of displays draw the locations of the DM windows in different places, but otherwise, the files are similar.

```
# STARTUP, /SYS/DM, default system startup command file for 1280x1024 monochrome
#
# Default is black characters on a white (or green) background.
INV -ON
# Window positions for the DMs input and output windows.
# Do not comment these out.
(692,1011)dr;(1279,1023)cv /sys/dm/output
(0,1011)dr;(639,1023)cv /sys/dm/input
(640,1011)dr;(692,1023)cv /sys/dm/output;pb
```

Figure 3–6. Start–Up Script for DN3000 Monochrome (startup.1280bw)

3.9 Server Process Manager Startup

If you start SPM on a node with a display, be aware that during the SPM startup, the '**node_data/startup.spm** file will be executed. If this file contains the same commands as the DM start–up file, you will execute the same commands twice. If you regularly start

SPM from a node with a display, ensure that all the node start-up commands are in the **/etc/rc.user** file and delete the **startup.spm** file from the node.

If SPM is running on a node (with or without a display), you can use the **crp** command from remote nodes to start processes on the node running SPM.

Although it is not recommended, you can start server processes in the SPM start-up file. The processes will run regardless of the node's log-in and log-out activity and will run as **user.server.none**.

Using 'node_data/spm_control to Control Node Access

The **'node_data/spm_control** can prevent unauthorized users from creating processes on or logging into a node. If this file exists on the node running SPM, all process creation and login requests are validated. Only users with a SID matching an entry in the file are allowed access; all others are rejected. If the file does not exist, all requests are allowed.

Each SID entry should be in the following format:

person.group.org

where a percent character in a field matches anything.

Figure 3-7 shows a sample **'node_data/spm_control** file.

```
# allow access to all users
%.%.%
# allow access to all members of group grp
%.grp.%
```

Figure 3-7. Sample 'node_data/spm_control File

SIO Line and Window System Startup

SIO lines connect "dumb" terminals to workstations, either directly or through a modem and telephone lines. The **/etc/ttys** file contains commands to start SIO lines and window systems. A sample line from the **/etc/ttys** file is shown below:

```
tty01    "/etc/getty d1200" dialup    on        secure
```

The line starts SIO port 1, specifying that it will be monitored by the **getty** program, that it will run 1200 baud, and that only users with root access will be allows to access the line.

The format for the command lines in the **/etc/ttys** file follows:

tty_line command terminal_type status [**window="***cmd_string***"**]

tty_line	The terminal's entry in the device directory, **/dev**.
command	The command to execute for the line. This entry is usually **getty**. **Getty** is a program that performs such tasks as recognizing the baud rate, reading the log–in names, and calling **login**. For *command* you can also specify the start–up file for a window system terminal emulator or some other server process.
terminal_type	The type of terminal normally connected to that tty line, as found in the **termcap** database file.
status	A 2–entry option. For the first entry, specify **on** to enable the line; specify **off** to disable the line. For the first entry, to allow only root to log in on this line, specify **secure**. Do not quote this field.
window="*cmd_string***"**	*cmd_string* is the pathname of the window system to start. If you specify a window system, it is started before any command is run for that line.

Tabs and/or spaces separate the fields. Use double quotes to enclose fields that contain more than one word (except the *status* field). If you omit a field, its value defaults to null.

NOTE: Previous releases used the SIO line servers **siomonit** and **siologin** to enable SIO lines. Although we recommend that you use the **/etc/ttys** file to enable SIO lines, you can still use **siomonit** and **siologin**. To do so, disable all the entries for serial lines in **/etc/ttys** by setting their *status* to **off**. Then, see "siomonit and siologin Line Servers" at the end of this chapter for information about using **siomonit** and **siologin**.

3.9.1 User Login

Figure 3–8 illustrates user log–in processing. The text following the figure describes each step in the process.

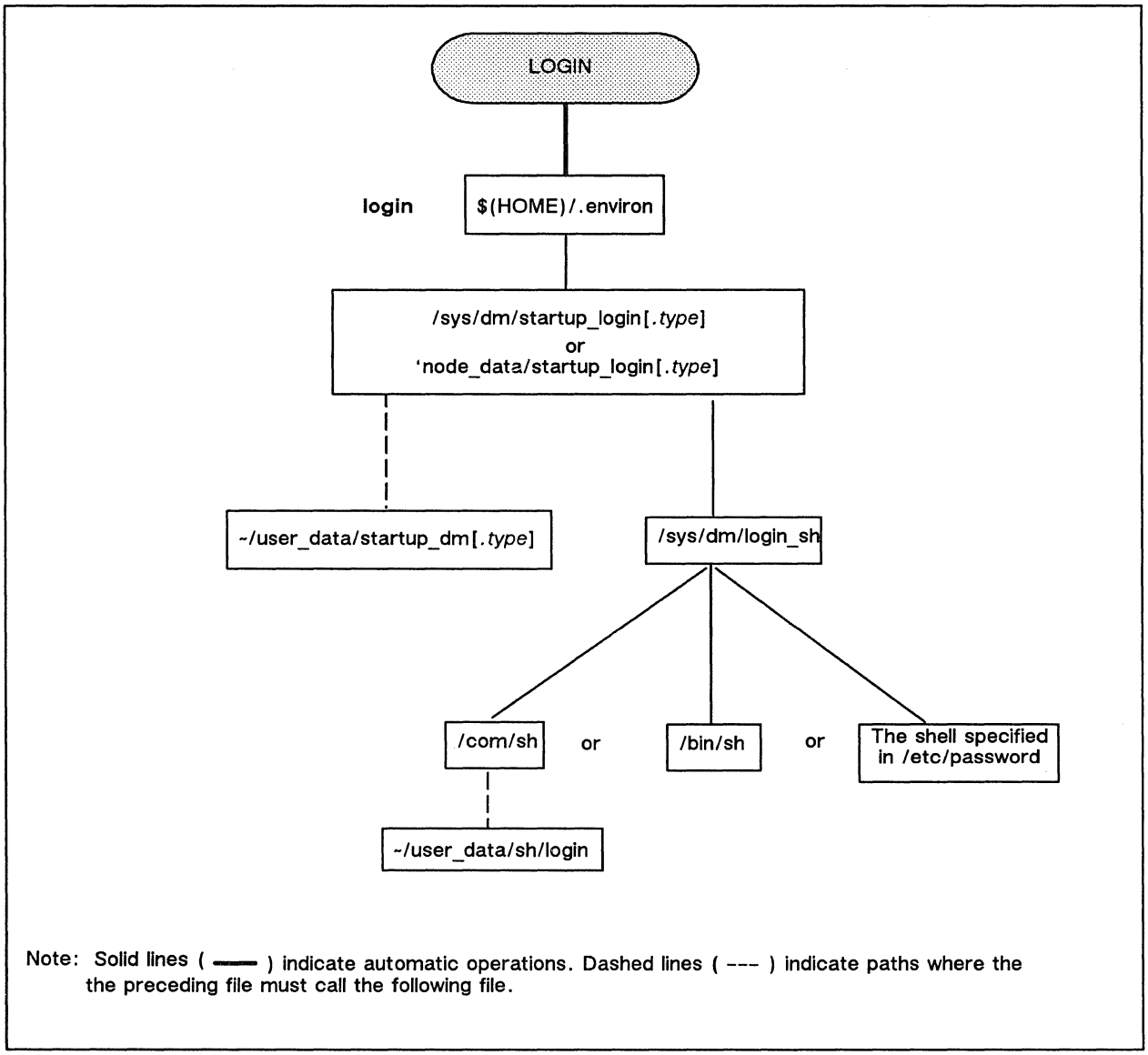


Figure 3–8. Node Start–Up Files and Operations

User Log–In Processing

1. When a user logs in to the node, the system:

   — First looks in the user's home directory for the **$(HOME)/.environ** file. If this file exists, it sets the environment and systype accordingly (if the corresponding environment was installed on the node).

   — Looks for a shell definition in the user's registry entry. If one is not there, the appropriate shell for the current environment is used.

   — Loads the key definitions based on the current environment.

2. The DM executes one of the following files:

   — **'node_data/startup_login**[.*type*]. The DM first looks for this file. If it finds it, it executes the file.

   — **/sys/dm/startup_login**[.*type*]. If the DM cannot find the file in **'node_data**, it uses this one.

   The **startup_login** file is used to perform tasks that need to be invoked every time someone logs in to this node. These tasks include, specifying windows, creating shells, and running user-specific scripts. The **/sys/dm/startup_login** file is supplied with the system. You should copy it to each node's **'node_data** directory. Then you can modify it specifically for each individual node.

3. As supplied, the **startup_login** file invokes **-/user_data/startup_dm**[.*type*]. This file, which contains commands private to the user to be executed at his login, is not supplied with the system, but must be created. To cause it to be executed, remove the pound sign (#) from the last line in the **/sys/dm/startup_login** file. (The line reads: **# cmdf user_data/startup_dm**[.*type*]). The SPM does not execute this script during remote logins. See *Using Your BSD Environment* for more information about this start-up file.

For the Aegis shell only (**/com/sh**), users can create a file of shell commands (**-/user_data/sh/login**) to be executed upon login. For more information about this file, see *Using Your BSD Environment*.

Key Definitions

When a user logs into the node, the DM sets the default key definitions for the node according to the system type specified in **/etc/environ.** or (if one exists) **$(HOME)/.environ.** Three sets of standard key definitions exist, one for each environment. When the user logs out, the DM resets the key definitions to the system type specified in **/etc/environ.**

3.9.2 Log-Out Script Processing

The DM processes log-out scripts. The log-out script must exist in either the **/sys/dm** or the **'node_data directory**, and the script must be named **startup_logout**[.*type*], where .*type* is the appropriate display type suffix. You cannot start up new processes with the DM **cp**, **cps**, or **cpo** commands from this script.

3.10 Start-Up File Summary

Several types of start-up command files exist: files executed by the operating system at boot time, files executed by the DM and the SPM, and user files executed at log-in time, or when a shell is started.

Table 3-3 lists the command files that can be used at these times. Some of these files run automatically; others must be specified in start-up files. The following subsections describe

the files that execute when the DM or SPM start running and when you log in to the DM. *Using Your BSD Environment* describes the shell start-up files and their functions in detail.

Table 3-3. Start-Up Files

File	When Run	Comments
/etc/rc	System boot	Runs automatically
/etc/rc.local	System boot	Must be specified in /etc/rc
/etc/rc.user	System boot	Must be specified in /etc/rc
/sys/node_data[*.node_id*]/startup[*.type*]	DM startup	Runs automatically
/sys/node_data[*.node_id*]/startup.spm	DM or SPM startup	Runs automatically on DSPs
/sys/dm/startup_login[*.type*]	User login	Runs automatically
/sys/node_data/startup_login[*.type*]	User login	Runs automatically
/dm/login_sh	User login	Runs from /sys/dm/startup_login [.type]
~/user_data/startup_dm[*.type*]	User login	Must be specified in startup_login
~/user_data/sh/login	User login	Runs automatically
~/user_data/sh/startup	By /com/sh	Runs automatically

NOTE: A *.type* argument in a pathname represents a node display type identifier; A *.node_id* suffix to **/sys/node_data** represents the hexadecimal identifier of the diskless node for which the directory holds information.

3.10.1 Node Display Types and Their Start-Up Files

Because display formats differ, particularly in their pixel dimensions, different nodes require different information in their start-up files. For example, the length of the DM windows vary among display types. For this reason, each node and DM log-in start-up file has several different versions, one for each display type. The display type is indicated by a suffix added to the start-up file name. Therefore, **/sys/node_data/startup.19l** is the DM start-up file for nodes with an 800-pixel by 10242-pixel landscape display. Table 3-4 lists the display-type suffixes and indicates the node models to which they apply.

Table 3–4. Start–Up File Suffixes

File Suffix	Node Types
.spm	Server nodes
.1280bw	Monochrome DN3000 and DN4000
.1280color	DN580
.19l	DN300, DN320, DN330, DN460, DN550, DN560, DN570, Color Domain 3000 and Domain 4000
.color	DN660

The installation procedures copy all versions of each start–up file on a node. This procedure ensures that any node can be a partner for any type of diskless node. Similarly, it is useful to have multiple versions of the DM log–in files if you are likely to log in on nodes with different display types.

3.10.2 Templates for Start–Up Files

The installation procedures automatically create several start–up template files. Table 3–5 lists the files.

Table 3–5. Start–Up File Templates

File	Comments
/sys/dm/startup_templates/startup[.type]	Copied to **/sys/node_data.***node_id* for diskless nodes
/sys/spm/startup_templates/startup.spm	Copied to **/sys/node_data.***node_id* for diskless nodes
/etc/templates/rc	Copied to **/sys/node_data/etc** for diskless nodes
/etc/templates/inetd.conf	Copied to **/sys/node_data/etc** for diskless nodes

The files in the **/sys/dm/startup_templates** and **/sys/spm/startup_templates** directories serve two purposes:

- They are master copies of the DM and SPM start–up files. You should edit these master copies only if you are making changes that should propagate through to all nodes in the network and to all nodes to be added in the future. If you are changing the start–up files for individual nodes, edit the copy created for the specific node, not the master copy.

- They are used as the source for the files copied by **netman** whenever it creates a **/sys/node_data.***node_id* directory for a diskless node. For more details on this see Section 3.11, "Administering Diskless Nodes."

3.10.3 Start–Up File Format

The start–up files contain DM or shell commands. If the file is a shell script, it normally starts with one of the following lines to specify the shell that will interpret the commands:

#!/bin/ksh For the Korn shell

#!/bin/sh For the Bourne shell

#!/bin/csh For the C shell

The default versions of the start–up files that are created when you install the BSD software contain most of the commands that you are likely to need. However, most of these commands are commented out in the files with a pound sign (**#**). You must delete the pound sign from the start of the line to allow the command to execute. For example, to allow the **netman** process to run whenever the node boots, uncomment the following line in the appropriate start–up file by deleting the pound sign:

```
# cps /sys/net/netman
```

NOTES: The DM, the Server Process Manager (**spm**), and the shell are all case sensitive. Use lowercase characters when referring to system commands and files.

Changes that you make in a start–up file do not take effect until the next time the file is run. For example, changes to the DM start–up files do not take effect until the next system boot. To start a process before the next operating system boot, use the **cp**, **cps**, or **cpo** command.

3.11 Administering Diskless Nodes

There is no apparent difference between working on a diskless node and a disked node. However, before you can use a diskless node, you must configure the node and start the processes that support the diskless node's operation. The following subsections describe the rules and techniques for managing diskless nodes and their partners. The last subsection describes a procedure for configuring a diskless node's partner.

3.11.1 Diskless Node Operation

When a diskless node displays the log–in prompt, all the programs required for its operation are in place. While *Using Your BSD Environment* gives a complete description of diskless node bootstrap operation, the following summary indicates what happens after you power on a diskless node in NORMAL mode.

1. The diskless node's Mnemonic Debugger broadcasts a message requesting a volunteer disked node partner.

2. Each disked node running the **netman** program listens for such a "request for volunteer" broadcast. The disked node answers the request if the requester's hexadecimal node ID is in the disked node's **/sys/net/diskless_list** file.

3. The diskless node loads **netboot**, its version of the operating system boot program from the partner, and proceeds with the bootstrap operation.

4. If the *//partner*/sys/**node_data.***diskless_node_id* directory does not exist, for example, if the diskless node has never booted from this partner, the **netman** program creates the directory and copies the node **startup**[*.type*] file from the **/sys/dm/startup_templates** directory.

5. The diskless node's DM executes the commands in the *//partner*/sys/**node_data.***diskless_node_id*/**startup**[*.type*] file.

6. The diskless node runs in the same manner as a disked node, using the partner node's disk for its system software.

3.11.2 Establishing Diskless Nodes and Their Partners

A diskless node's partner node provides the system software and disk services for the diskless node. The partner does not necessarily store any of the diskless node user's files. Each partner node must run the diskless node server, **netman**. Partners can be nodes with, or without displays. Nodes without displays are DSPs.

A partner node must have the correct system software for the diskless node type. For example, if the diskless node is a DN570 and the partner node is a DSP90, the partner node must have both a **/sau3** directory and a **/sau5** directory. Similarly, the partner's /sys directory must have any microcode files required by the diskless node.

Each diskless node has its own /sys/**node_data.***diskless_node_id* directory on the partner node. The '**node_data** directory on each diskless node resolves to the directory /sys/**node_data.***diskless_node_id*. If you change diskless node partner assignments, delete the /sys/**node_data.***diskless_node_id* directory from the original partner's /sys directory.

The home directories of diskless node users can be located on any node in the network, and do not have to be on the diskless node's partner node. Whenever possible, locate the home directories on the same loop as the diskless node. If a diskless node has one or more regular users, their personal log-in start-up scripts, **user_data/startup_dm[.***type***]**, should be in their home directories.

Specifying Partners

A disked node can be a partner to one or more diskless nodes. You control the assignment of diskless nodes to partners through the disked node's /sys/**net/diskless_list** file, such as the one shown in Figure 3-9. The disked node will volunteer to be a partner for any diskless node whose node ID is in the disked node's **diskless_list**.

```
# This is the diskless list for the network server on node 6b2d.
#
# The first token in each line of this file is examined by netman
# when it receives a network bootstrap volunteer request. If the
# token is a valid node ID, then netman will volunteer to help
# that node when it calls. Lines that do not begin with a valid
# node ID will be ignored. The use of the comment line
# character "#" is recommended.
#
# The nodes that this file authorizes netman to volunteer help for
# are:

3f4
eff21
4d76
```

Figure 3-9. A Sample /sys/net/diskless_list File

Choose the partners for diskless nodes carefully. For example, a partner should be in the same network loop as the diskless node; then, if you switch the loop out of the rest of the network, the diskless node can still function.

NOTES: If you put a diskless node's node ID on more than one diskless list, you cannot predict which node will become the partner when the diskless node boots. As a result, the diskless node's **'node_data** directory and the contents of that directory, could change whenever the node reboots. Therefore, you must configure the diskless node correctly on each possible partner node.

If you use names for diskless nodes, do not change the name of the /sys/**node_data.***diskless_node_id* directory to /sys/**node_data.***diskless_node_name*. Remember, a diskless node name is *not* valid in a pathname. You must specify the diskless node ID to accurately access this object.

When you boot a diskless node, you can request a specific partner node. See "Requesting a Specific Partner," later in this section.

The /sys/node_data.node_id Directory on New Partners

If a diskless node's partner does not have a **/sys/node_data.***node_id* directory when the diskless node boots using the partner, **netman** automatically creates one. If this directory does not contain the minimal set of files required by the diskless node to operate, **netman** creates them. The **netman** process also copies the **startup[.***type***]** file from the partner's **/sys/dm/startup_templates** or **/sys/spm/startup_templates** directory.

Providing a New Partner for a Diskless Node

Use Procedure 3–1 to configure a partner for a new node, change a partner for an existing node, or to provide an additional partner for an existing node. If a service representative installs a diskless node, he or she creates a partner for the node; you might want to use a different partner.

PROCEDURE 3–1. *Providing a Permanent Partner for a Diskless Node*

Task 1: Determine the diskless node's ID

If the node is new, the node identification slip lists the node ID.

If the diskless node currently has a partner, you can determine the ID by entering the following command at the diskless node.

```
% netstat
The node ID of this node is eff21.
```

Task 2: Log in to the partner node

If the partner has a display, log in at the partner node.

If the partner does not have a display, you can create a remote shell on the partner by using this command:

% crp /bsd4.3/bin/start_sh -on *//partner* **-me**

Task 3: Add the diskless node to the partner's list of authorized nodes

Edit the partner node's **/sys/net/diskless_list** file, and add the diskless node's node ID to the list.

If you are changing partner nodes, delete the diskless node's ID from the old partner node's **/sys/net/diskless_list** file. Also delete the old partner's **/sys/node_data.***diskless_node_id* directory.

Task 4: Start netman on the partner node

If the partner node is running **netman** already, go on to Task 5.

If the partner node is not running **netman**, do the following:

1. Remove the pound sign (#) from the following line in the partner's appropriate start–up file:

   ```
   # cps /sys/net/netman -n netman
   ```

   The **netman** server will now start automatically whenever the partner reboots.

2. Start the partner node's **netman** server. (By doing this, you do not have to reboot the partner.)

   If you are at the partner node, enter the following command at the DM command prompt:

   Command: **cps /sys/net/netman**

   To start **netman** from a remote node, enter the following command. (You must use this command even if you used the **crp** command in Task 2.)

   % crp -on *node* **-cps /sys/net/netman -n netman**

Task 5: Limit the size of the partner's memory pool (optional)

You can limit how many memory pages are left available for diskless node paging requests using the **netsvc** command. (Be aware, however, that this command affects the memory pool size for all situations in which files are used remotely on your node.)

To limit the size of the partner's memory pool, type:

% netsvc -p [*pool_size*]

where *pool_size* is the maximum number of memory pages that will be available for diskless node paging.

If you don't use this command, all of the partner's memory is available for paging requests from the diskless node.

Task 6: Name a diskless node (optional)

If you are installing a new diskless node, you can give it a name. Skip this step if the node already has a name.

If you do not use **ns_helper** on your network, enter the following command.

% ctnode *node_name node_id*

If you do use **ns_helper** at your site, enter the following:

% ctnode *node_name node_id* **-root**

After you finish this procedure, catalog the node on the network.

Task 7: Create node start-up and configuration files

1. Create the diskless node's **/sys/node_data.***node_id* directory by entering the following command at the partner node:

   % mkdir /sys/node_data.*node_id*

   where *node_id* is the diskless node's ID.

   The **netman** process will copy the **/sys/node_data/startup***[.type]* file from the partner to the diskless node's **/sys/node_data.***node_id* directory (where *type* is the type of display on the diskless node, not the partner). **Netman** copies the file from the partner's **/sys/node_data** directory, or from the **/sys/dm/startup_templates** directory (for diskless nodes with displays), or from the **/sys/spm/startup_templates** directory (for diskless DSPs).

NOTE: If you are changing the partner of an existing diskless node, **netman** copies these configuration files from the old partner to the new partner.

2. Edit the **/sys/node_data.***node_id***/startup[.***type***]** and **/sys/node_data.***node_id***/etc/rc** files to configure the diskless node.

Task 8: Log off the partner node

Task 9: Reboot the diskless node

You can now start or restart the diskless node.

If you are changing an existing diskless node's partner, enter the following DM command to log off and shut down the operating system.

```
Command: shut
```

The Mnemonic Debugger prompt (>) appears on the screen. Enter the following reset command:

```
> re
```

Press <RETURN> twice.

The PROM identifier appears, followed by the Mnemonic Debugger prompt. Restart the operating system by entering the following command:

```
> ex domain_os
```

The node now restarts, using the new partner node; you can now log in.

3.11.3 Managing Diskless Nodes and Partners

You should evaluate your diskless node partner assignments from time to time. Distribute assignments in a way that does not degrade the performance of either partner. The **netmain_srvr**, the network maintenance server, and the **netmain** interactive tool, along with the **netsvc** shell command, can help you to manage partner assignments.

Diskless Node Management Commands

The **netmain_srvr** collects information about the number of diskless nodes assigned to any partner. The **netmain** interactive tool formats the performance data so that you can identify nodes providing more than their share of resources to diskless nodes in the network. For a detailed description of **netmain_srvr** and **netmain**, see Appendix A.

Sometimes the partner node is referred to as the paging partner because it performs the paging services for the diskless node. The diskless node copies information (operating system, global data, etc.) in 1024–byte pages from its partner as needed.

Whenever a diskless node cannot communicate with its partner, it displays a message to that effect. You will see this message if the partner stops running the operating system because of an intentional shutdown or system crash. Problems with the network can also cause the diskless node to display the message. Once the partner node is rebooted or the network is back up, use CTRL/F to refresh the screen on the diskless node and eliminate the messages. You must reboot the diskless node whenever its partner node reboots.

Warning of a Partner Shutdown

It's good practice for the administrator of a partner node to notify diskless node users of intended shutdowns. Use the **send_alarm** or the **/etc/wall** command to notify users of shutdowns. For example, if your are shutting down your disked node, the following command will warn all diskless nodes that use your nodes as a partner:

```
% send_alarm 'Partner node disked is shutting down in 2 min. Please log out.' -mydi
```

Requesting a Specific Partner

If for some reason a diskless node's regular partner is not available, the diskless node can request as its partner another disked node that runs **netman**, even if that node does not have the diskless node on its partner list. You can also use this procedure to boot a disked node as a diskless node, which you might need to do if, for example, the node's system software is corrupted or if you are initializing its disk across the network. This is a temporary measure, however; you should not use this procedure in place of the **diskless_list** file.

Use the following steps if you must boot a diskless node by requesting a specific partner:

1. Boot the diskless node in SERVICE mode or, if the node is running the DM, use the DM **shut** command to enter the Mnemonic Debugger.

2. Enter the following command when the Mnemonic Debugger prompt (>) appears:

   > di n *node_id*

   where *node_id* is the hexadecimal ID of the partner node you are requesting.

3. Enter the following command to restart the node:

   > ex domain_os

3.12 Node Troubleshooting

If a node doesn't appear to be operating correctly in the network, you can try to diagnose the problem in a number of ways. This section does not attempt to be exhaustive, but it does outline some common faults that occur and how to correct them.

Check Connections and Power

Check the brightness control on the monitor. Make certain that cables are attached correctly and securely. Make sure the power is on.

Check LEDs

If the node has LEDs (such as the DN3000 series), see if they're active. If they are, suspect problems outside the node, perhaps with the network.

Try to Communicate with the Hung Node

Try to list the entry directory of the hung node. For example, if the node name is chinook, try the following command line from another node on the same network.

% **ls //chinook**

If you receive the message "name not found," try recataloging the node name, as shown in the example below, and then retry the listing. If you receive an "object not found" message, make sure you're not on a loop that's been switched out of the network.

% **ctnode** *hex_id* **chinook -r -l**

Is the hung node running diskless? If so, the problem might be that the partner node has crashed, or that temporary network problems have disturbed the communication between the partner and the diskless node. Remember, too, that a diskless node can have a name, but doesn't have an entry directory, so if you **ld** a diskless node name as if it were an entry directory (for example, **//diskless**), you will receive the message "object not found."

Check Processes

You can check the process running on the node in one of two ways. Either use the **crp** command to log in to the node and run the **ps** or **dspst** process status commands, or use the **-n** *node_spec* option with either the **ps** or **dspst** command to look at the processes. Look for exceptionally heavy network traffic or for a process taking up an inordinate amount of CPU time.

3.13 Server Reference Information

This section describes the following servers:

- Alarm Server — **alarm_server**
- Mailbox Server — **mbx_helper**
- Diskless Node Server — **netman**
- Tablet Server — **spb1**
- Serial I/O Line Servers — **siologin** and **siomonit**
- Clock Server — **cron**
- Remote User Communication Server — **talkd**
- Write Server — **writed**

You can obtain online help for any server by typing the following line, substituting the name of the server for *server_name*.

% **help** *server_name*

Appendix A provides reference information on the Server Process Manager and the **netmain** server.

3.13.1 The Alarm Server: alarm_server

The Alarm Server (/sys/**alarm/alarm_server**) alerts you by popping a small alarm window on the screen and sounding an alarm.

Run the Alarm Server as a background process by starting it with the DM command **cpo**. Usually, you'll start the Alarm Server from a start-up file.

The Alarm Server can report on potential disk overflow, network problems, **netmain_srvr** observations, and brief messages from other users. Each condition is checked every four minutes by default, or at an interval that you set with the **-period** option.

By default, an alarm is accompanied by a distinctive tone pattern. You can disable the audible alarm or have all alarms alert you with a single short beep.

Alarm messages appear in windows of default sizes, accompanied by standard sound patterns. The first alarm window appears near the top left-hand corner of the screen. Use command options to alter the default window positions. A "pad closed" message appears at the bottom of the alarm window when the alarm is complete. Note that if you inadvertently press CTRL/Q in the window before the message appears, you will kill the Alarm Server.

Certain optional software packages also use the Alarm Server. See the release notes and documentation for any optional software you have purchased for details about the alarm server's operation with those products.

Starting the Alarm Server

To start the Alarm Server, enter the following from the DM command line:

```
Command:  cpo /sys/alarm/alarm_server  -[options]
```

The server process begins immediately and ends at logout.

To start the Alarm Server from a start-up file, include the following line in the file. You'll have to log out and log in again for the server to start.

```
cpo /sys/alarm/alarm_server  -[options]
```

The Alarm Server names itself "alarm_server" by default, so you don't need to specify the **-n** option with the **cpo** command.

Configuration Files

To execute Alarm Server options from a configuration file, create a file using the format illustrated in Figure 3–10.

```
-disk 75
-hw
-nm_srvr //netmain_srvr_node
-bell1
```

Figure 3–10. Sample Alarm Server Configuration File

Edit the appropriate start–up file in the **~/user_data** directory to specify the configuration file. For example, in the user's home directory start–up file, the command

```
cpo /sys/alarm/alarm_server '*~/user_data/options' -n alarms
```

uses the standard command line option, the asterisk (*), to point to the configuration file. The command names the process "alarms."

Alarm Server Options and Arguments

To receive alarms about any of the standard Alarm Server events, specify the event with one or more options described below. The default event reports are indicated by (D).

-disk[_full] [*nn*]

Posts an alarm when the disk containing the node's "/" directory is more than *nn* percent full. If you omit *nn*, the **alarm_server** uses a default value of 95 percent full (5 percent free space). If you do not delete anything from the disk, or if the full–disk condition recurs, **alarm_server** notifies you again. After two notifications of a full–disk condition, **alarm_server** does not notify you again for at least one hour, even if the condition persists.

Default if omitted: Do not post disk–space alarms.

-hw[_fail]

Posts an alarm when some node on the network detects network hardware problems (as seen in the "last ring hardware failure" report from the **netstat -l** command). The **netstat** output lists the last hardware failure report detected on the network, whether it is new or not. The Alarm Server posts alarms only when there is a new report, indicating a current problem.

Default if omitted: Do not post a network problem alarm when there is a new hardware failure report.

-netmain [*pathname ...*]

Enables alarms from **netmain_srvr** observers. The *pathname*(s), if specified, represent text files containing lists of nodes that run **netmain_srvr**. If you don't specify a file for *pathname*, the Alarm Server uses the file **~/user_data/alarm_server.netmain_srvr_list.**

The files should contain node names or hexadecimal node ID numbers on different lines or separated by spaces on the same line. Comments in these files start with a left brace ({) or a pound sign (#) and run to the end of the line.

The Alarm Server reads these files only when it starts up. If you add or delete node names in these files after the server is running, changes do not take affect until the next time the server starts up.

Default if omitted: Do not enable alarms from **netmain_srvr** observers on node lists.

-nm_srvr *node_spec* [...]

Enables alarms from **netmain_srvr** observers on the node(s) specified with *node_spec*, which is a hexadecimal node ID or node name.

Default if omitted: Do not enable alarms from **netmain_srvr** observers specified in the command line.

-nonetmain (D)

Prevents the Alarm Server from checking for observer alarms from the **netmain_srvr** program.

-nets

Notifies you when a network disappears from your internet or appears in your internet. This option is useful only in Domain internet environments.

-nonets (D)

Suppresses alarms describing changes in the internet topology.

-msg (D)

Allows the alarm server to receive messages from the **send_alarm** program.

-nomsg

Prevents the alarm server from receiving **send_alarm** messages.

-bell1

Sounds a single short beep for any alarm, instead of the usual distinctive tone pattern for this alarm type. The **-nobell** option overrides **-bell1.**

Default if omitted: Use distinctive tone patterns for each alarm type.

-nobell

Suppresses audible alarms for all alarm types.

Default if omitted: Sound audible alarms.

-p[eriod] *nnn*

Checks each alarm detector every *nnn* minutes, where *nnn* is a decimal number greater than or equal to 1.

Default if omitted: Check each alarm detector every four minutes.

-v[ector] *dx* [*dy*]

Separates alarm windows by *dx* and, optionally, *dy*, where *dx* is the difference (in pixels) between the horizontal coordinates of each alarm window and *dy* is the difference between the vertical coordinates. Specify *dx* and *dy* as decimal integers. For example, you might want the top left corners of successive alarm windows to be 0,0 for the first window, 20,20 for the second window, and so on; to do so, use the option -v 20 20. (Use the -w option to specify the location of the initial window.)

Default if omitted: **vector** 235 0

-w[indow] *initx* [*inity* [*width* [*height*]]]

Sets the screen position of the first alarm window and the size of the alarm windows.

Default if omitted: **window** 1 1 225 100

Examples

The following example causes the Alarm Server to post a nonaudible alarm when the disk is 98 percent full (2 percent free space):

```
% cpo /sys/alarm/alarm_server -disk 98 -nobell
```

The following example causes the Alarm Server to post alarms when the disk is 90 percent full, when there is a new hardware failure report message, and whenever there is an observer report from the **netmain_srvr** running on node ID "ffff5." The command also sets the screen position of the first alarm window position to be the upper left-hand corner of the screen.

```
% cpo /sys/alarm/alarm_server -disk 90 -hw -nm_srvr ffff5 -w 5 5
```

This next example make use of an options file that you've created (in this case **-/user_data/opts**) to control the Alarm Server. This command also uses the **-n** option to name the Alarm Server process "alarms".

```
% cpo /sys/alarm/alarm_server '*-/user_data/opts' -n alarms
```

Special Considerations

Occasionally , when the Alarm Server starts, it prints a warning message about a file called **.../alarm_server.msg_mbx**. Such a message can come from several sources. Often the problem is caused by incorrect protection about the mailbox used by the server process (see **mbx_helper**). The Alarm Server can usually change the protection on its message mailboxes automatically, but it will sometimes need your help.

The protection on two files must allow the **mbx_helper** program Read (r) and Write (w) access. Make sure that the files '**node_data/alarm_server.msg_mbx** and ~/**user_data/alarm_server.msg_mbx** both allow Read and Write access to user.server.none.

Related Information

Information about using the Alarm Server with **netmain_srvr** is provided in the section on the **netmain_srvr** process in Appendix A.

Additional information about using the Alarm Server is available with certain optional software packages. For example, the Domain Software Engineering Environment (DSEE™) is an optional software package that uses the Alarm Server. If DSEE is installed on your system, see the DSEE manuals for additional information.

3.13.2 The Mailbox Server: mbx_helper

The Mailbox Server (/sys/**mbx/mbx_helper**) assists processes on different nodes that communicate using mailboxes. This server must run on any node that sends or receives mailbox communications across the network.

The **mbx_helper** process is automatically started by any Apollo servers or programs that need the server to operate correctly. To send a message via the **send_alarm** command, both the sending and receiving nodes must have an **mbx_helper** process running. For the sending node, if it doesn't already have an **mbx_helper** running, the **send_alarm** command will start one. At the receiving node, the **alarm_server** process will start an **mbx_helper** process if one doesn't already exist there.

You only need to start an **mbx_helper** explicitly, with a DM **cp** command or from a start-up script, if you are running a non-Apollo server that requires **mbx_helper** to operate.

For further information about starting **mbx_helper** from programs, see *Programming With Domain/OS Calls*. To provide network services to a node, enable **mbx_helper** by removing the pound sign (#) from the appropriate start-up file on that node.

Special Considerations

In a secure network, a mailbox receives its permissions from the directory in which it is created. The ACL templates we supply ensure that server processes can have access to each other. If you do not use the templates we supply, be certain that the permissions you use allow clients on remote nodes to access a node's **mbx_helper**. If user programs have difficulty accessing **mbx_helper**, be certain that users know how to use **mbx_helper** in a secure network. See *Programming With Domain/OS Calls* for more information.

You cannot change a mailbox's permissions while the mailbox is in use.

3.13.3 The Diskless Node Server: netman

The **netman** (/sys/**net/netman**) process manages requests from diskless nodes for access to the operating system. Diskless nodes, having no place to store the necessary operating system files, use a disked node's disk for storage.

The **netman** process, running on a disked node, receives "request for volunteer" broadcasts from diskless nodes attempting to boot. **The netman** server looks in its **/sys/net/diskless_list** for a diskless node's hexadecimal ID. If the ID appears in the list, the diskless node can read, from netboot, the disked node on which **netman** runs. To control the distribution of disked node resources in the network, specify which diskless nodes can use a given disked node as a partner. Do this by placing diskless node IDs in the partner's **/sys/net/diskless_list**.

If a diskless node does not have its own '**node_data.***diskless.node_id* directory when it boots, **netman** will create one for it from a template in the disked node's **/sys/dm/startup_templates** directory. See Section 3.8 for more information about node start-up directories.

The **netman** server runs as a background process from a start-up file. The command line that executes **netman** is in the start-up files that arrives with your system. Remove the pound sign (#) at the beginning of the command line to enable the process. The **netman** server will execute from the start-up script when the node is rebooted. You may start the process from the DM command line as shown in the next section.

> **NOTE:** We do not recommend that you run **netman** on an internet routing node. Network traffic between a diskless node and the routing node would compete with the internet traffic on the routing node, slowing both the internet traffic and the response time of the diskless node.

Starting and Stopping netman

To start **netman** from the DM command line, enter the following command:

```
Command: cps /sys/net/netman
```

The server process begins immediately and persists after logout. Another way to start **netman** is to uncomment the following line in the appropriate start-up file for the disked node:

```
# cps /sys/net/netman
```

The server process begins when the node is booted, and continues under normal conditions until it is intentionally stopped with the shell command **sigp**, as shown:

```
% sigp netman -q
```

With both start-up methods described above, the process stops running if the node is shut down intentionally or because the system crashes. Restart the process if it stops running by rebooting the node, or by entering the **cps** command from the DM command line.

Special Considerations

For a temporary fix, you can use **netman** if you disked node malfunctions. To do this you must shutdown and reboot your node diskless. The **netman** process allows any disked node to continue functioning even if its disk becomes nonfunctional.

For this temporary procedure:

1. Determine the node ID of any node running **netman**. (You may want to start the **netman** process on a disked node in the same loop as the now diskless node.)

2. Use the DM command **shut** on the node that you want to boot diskless.

3. Then type the following at the Mnemonic Debugger prompt:

```
> re

> di n node_id  {node_id is the node ID of the node running netman}

> ex domain_os
  Network Partner ID nnnn
```

In this procedure, **netman** bypasses **/sys/net/diskless_list**. You don't need to edit the list to get the node back in the network. When you use this procedure, **netman** creates a **'node_data/startup[.***type***]** file for the node with disk problems.

3.13.4 The Tablet Server: sbp1

The Tablet Server (**/sys/dm/sbp1**) supports tablets that conform to the binary output mode used by Summagraphics Corporation tablets. To use a tablet, enable the Tablet Server support program in the appropriate start-up file for the node. The server process sends a control character to the tablet bit pad. An operating system program then provides the actual bit-pad support. The process started in the start-up file stops running when the operating system has taken control.

Starting the Tablet Server

To start the Tablet Server, uncomment the following line in the appropriate start-up file:

```
# cps /sys/dm/sbp1 /dev/sio2 1
```

To use a different SIO line, change the "2" in the command to the desired number. The letter "1" indicates the mode and sampling rate selector that is sent to the tablet. You may change this mode to one of the other modes described in the *Summagraphics Corporation Bit Pad One User's Manual* (Form 64).

Special Considerations

The bit-pad support is internal to the operating system, and the server mentioned here only enables the operating system support and then stops. Therefore, the **ps** command does not show a process for the bit pad. To check on a bit-pad process, use the **tctl** command on the SIO line to which the bit pad is connected, and check that **-bp_enable** is true. The bit-pad support program is running properly if the **tctl** output contains the following line:

```
bp_enable: true
```

3.13.5 The siologin and siomonit Line Servers

Previous releases used the **siomonit** and **siologin** SIO line servers to allow you to connect a "dumb" terminal to a workstation, either directly or via modem and telephone lines from another location. Although we now recommend that you use the **/etc/ttys** file (see the "SIO Lines" section earlier in this chapter) to enable SIO lines, you can also use **siomonit** and **siologin.**

The SIO line server processes are

- **siologin** (SIO line login) — **/sys/siologin**
  Invoked by **siomonit**. It must be a manager within the log–in protected subsystem. **Siologin** is the process that directly manages user login.

- **siomonit** (SIO process monitor) — **/sys/siomonit**
  Typically invoked as a background server process from a start–up file.

Below, we describe the procedure used to connect terminals to workstations. Before you can use this procedure, set up the necessary configuration files and enable the server processes **siomonit** and **siologin,** described in the two reference sections that follow this one.

To connect a node's SIO lines to a dumb terminal and/or modem, follow these steps:

1. Disable the SIO lines in the **/etc/ttys** file. To do so, specify **off** in the *status* field for all but the first entry in the **/etc/ttys** file.

2. Use the shell command **tctl** to display the SIO line configuration. The default values for the SIO lines are

   9600 baud
   No parity
   Eight bits per character
   One stop bit

3. Change the SIO line configuration to conform to the modem or terminal configuration parameters. This may be done in a shell command file run by **siologin** when it starts. Alternately, change the terminal or modem configuration parameters according to the manufacturer's instructions to conform to the SIO line configuration.

4. Enable the **siomonit** server with the proper configuration files and arguments. The **siomonit** process starts the **siologin** process.

5. Connect the terminal or modem to the SIO line. Use the cable and directions supplied with the terminal or modem. If you have connected a modem, the line parameters on the terminal and modem at the remote site must match those you specified at the Domain workstation. Connect the modem to the telephone. When you are ready to log in, signal the SIO line by typing <RETURN> on a locally connected terminal. From a remote terminal, dial the number of the phone line connected to the node's modem.

When you are finished using a local or remote terminal, type CTRL/Z (or the character you have defined as the EOF character using **tctl**) to end the **siologin** process. On a local terminal, you are disconnected. On a remote terminal, you are disconnected from the node and the phone connection is broken.

3.13.5.1 The SIO Line Log-In Server: siologin

The **siologin** process uses the following command-line syntax:

siologin *dev_name* **[[-dialin]** **[-n** *name*] *prog* [*argument...*]]

Each **siologin** server process waits for a carriage return character from a terminal connected directly to the SIO line, or a Data Carrier Detect (DCD) signal from a modem if the **-dialin** option has been specified. The modem generates this signal when it answers a dial-in from a remote terminal.

Upon receiving the character or signal, **siologin** invokes the operating system log-in sequence. If the sequence is successful, **siologin** logs the user in and starts a program. You specify the program with the *prog* specification. The default option starts the shell command line interpreter program **/com/sh**. The *dev_name* argument, which must be specified, is the SIO device descriptor pathname. Other options, if specified, must precede *prog* and its arguments. The **siologin** subsystem process stops when the user logs out (usually with CTRL/Z).

The **siologin** command-line syntax appears as an argument list in a file used by **siomonit** (usually **'node_data/siomonit_file**). We describe the meaning of **siologin** options and arguments below.

Siologin looks for a start-up file **'node_data/startup_sio.sh** and, if it exists, executes it as a shell command file, by passing it the SIO line number as an argument. For example, for **/dev/sio2:**

```
/com/sh `node_data/startup_sio.sh 2
```

The sample file, **/sys/siologin/startup_sio.sh**, provided with the system includes the **tctl** command shown below to to ensure that the SIO line is not locked through some previous failure:

```
# ulkob /dev/sio2 ^1 -f
```

The siologin Options and Arguments

The **siologin** process accepts the following arguments:

dev_name (required) — The SIO device descriptor pathname, in the form **/dev/sio***x*, where *x* is the number of the SIO line to which the terminal or modem is connected. Use SIO line numbers 1, 2, or 3 for nodes with three SIO lines; use SIO line numbers 1 or 2 for nodes with two SIO lines.

prog — A program for **siologin** to start after the login is complete. If omitted, the default invokes **/com/sh** to start the shell command line interpreter.

args — Arguments for the program specified in [*prog*].

The **siologin** process accepts the following options:

-dialin — The SIO line connection is remote. If the line is remote, **siologin** asks for an access password before invoking the log-in sequence. The access password is a single string read from **'node_data/siologin_access**. For remote lines, **siologin** waits for a carrier detect signal to initiate the operating system log-in sequence. It disconnects the line after the invoked program returns. (If the connection is local, **siologin** waits for a <RETURN> before beginning the log-in sequence.) **The siologin** process logs invalid log-in attempts in the file **'node_data/siologin_log.**

-n *name* — Specifies the name to give the **siologin** process. (Takes precedence over the option **cp -n** when **siologin** is created through the DM command instead of through the **siomonit** server process.)

Special Considerations

When you use the **-dialin** option to specify remote access, you must specify an access password in the file **'node_data/siologin_access**. Create the file by entering a password as a left-justified single string in the file's top line.

Siologin logs successful and unsuccessful log-in attempts from remote terminals. It creates the file **'node_data/siologin_log**, which reports the SIDs of users who log in successfully, and provides error messages describing unsuccessful log-in attempts. You should monitor **'node_data/siologin_log** and periodically delete it, or delete old entries, since it is not self-limiting in size. Figure 3-11 shows a sample log file.

```
Thursday, February 14, 1985  13:40:52
     ** Bad (or no) access code in `node_data/siologin_access **
 Thursday, February 14, 1985  13:40:52
     ** Hanging up phone **
 Tuesday, February 19, 1985  15:52:29
     Invalid login attempt by: jones
    Tuesday, February 19, 1985  15:53:37
     Invalid login attempt by: jones
    Tuesday, February 19, 1985  15:53:48
     jones.dev.mktg.5de logged in
 Tuesday, February 19, 1985  15:58:19
     Caller logged out.
```

Figure 3-11. Sample 'node_data/siologin_log File

To operate, the **siologin** server must have manager status in the login protected subsystem (see *Using Your BSD Environment* for more information about the subsystem.) The software installation procedures give **siologin** manager status. If the server does not operate properly when **siomonit** starts it, display its subsystem status as follows:

% subs /sys/siologin/siologin

If you receive the following message, **siologin** has the proper manager status:

```
"/sys/siologin/siologin" is a login subsystem manager
"/sys/siologin/siologin" is a file subsystem data object
```

If, however, you receive the following message, the **siologin** process has lost its manager status:

```
"/sys/siologin/siologin" is a nil subsystem manager
"/sys/siologin/siologin" is a file subsystem data object
```

To reassign manager status to **siologin**, you must log in with a **sys_admin** account and type the following:

% ensubs login
% subs /sys/siologin/siologin login -mgr
% CTRL/Z

3.13.5.2 The SIO Process Monitor: siomonit

Siomonit supports repeated logins over SIO lines, independent of any log-in or log-out activity at the node terminal. To enable **siomonit**, create a file that describe the attributes of each **siologin** manager that the **siomonit** server process should start. The **siologin** processes started by **siomonit** are called its child processes.

The file passed to **siomonit** contains argument lists. Each argument list has the form of the **siologin** command line described previously. The **siomonit** process invokes a separate **siologin** process for each argument list in this file. A maximum of three argument lists (one per SIO line) can be given. Comments can be included in the file with a pound sign (#).

Starting siomonit

To invoke **siomonit** from the DM command line, enter the following:

Command: cps /sys/siologin/siomonit siomonit_filename

The server process begins immediately and continues after logout.

We recommend this start-up method if you use **siomonit** or **siologin** only occasionally. This is also the way to start the process after you log in or if the process dies. Be sure the SIO lines are configured correctly by using the **tctl** command. Usually the *siomonit_filename* argument is **'node_data/siomonit_file**.

To start **siomonit** from a start-up file, uncomment the line that reads as follows:

```
# cps /sys/siologin/siomonit -n siomonit `node_data/siomonit_file
```

Be sure this line appears after any lines in the start-up file that set environment variables.

The server process begins when the node comes online and continues across logins and logouts at the node terminal. We recommend this start-up method if you use **siomonit** and **siologin** frequently.

Signaling the siomonit Process

Sometimes you will want to signal **siomonit** once you have started it. You might do this, for example, after you make changes to an argument list or if you add a new argument list to the file **siomonit_file.**

To make **siomonit** reread the argument file and execute the process again, signal **siomonit** with an asynchronous quit fault (**sigp -q**). This is the default option of the **sigp** command:

```
% sigp siomonit
```

The **siomonit** process goes back to its argument file and redoes whatever it finds there. Note however that **siomonit** will never stop an active child process for a given SIO line, even if you have changed the argument list for that SIO line. You must stop the child process. This will also cause **siomonit** to reread the **siomonit_file**. To stop **siomonit**, send it a stop fault (**sigp -s**).

Restarting siomonit

If the **siomonit** process stops running, restart it from the DM command line (or by rebooting the node). Check the **siomonit_log** file first, to determine why the process stopped.

Sample siomonit_file

The siomonit_file name argument lists take the form:

[-repeat] *siologin_arg_list*

The arguments in the siomonit_file are explained below.

-repeat	Configures **siomonit** to restart this **siologin** process after a user logs out. For example, when a user of an SIO line logs out, no one else can log in to that line unless this argument is in effect. It signals **siomonit** to restart the **siologin** process. This argument should be the first one given.
siologin_arg_list	The list of **siologin** arguments and options described in the section on **siologin**. Arguments are passed to **siologin** unvalidated; however, the first argument must be **/dev/sio***x*. **Siomonit** reads the argument file over again each time a child process stops, when a user logs out, or when it receives a quit fault.

Figure 3-12 shows a sample **'node_data/siomonit_file.** You can include comments by placing the pound sign (#) at the beginning of a line.

```
# sample file for using siomonit
#
#  Configure the sio lines and invoke siomonit with a line like this
#  in  `node_data/startup (or /sys/dm/startup):
#  cps /sys/siologin/siomonit -n siomonitor `node_data/siomonit_file
#
#  Put the sio line startup file in `node_data/startup_sio.sh.
#
#  and put a file like this one in `node_data for the node to be used.
#
#  watch sio lines 1 and 2 and keep them available for siologin:
#  line 1 is a dial up line:-repeat /dev/sio1 -dialin -n siologin1  /com/sh -f -c
user_data/startup_sh #     line 2 is a local connection:
-repeat /dev/sio2   -n siologin2_local  /com/sh -f -c user_data/startup_sh
#
# up to three siologin arg lists like the ones above may be included, one
# for each sioline (only two will work for DN300's).
#
# This file is re-read by siomonit each time it re-invokes an siologin
#  process or when it receives a signal via:    sigp siomonit
```

Figure 3-12. Sample 'node_data/siomonit_file File

For each argument it finds in the list, **siomonit** invokes the **siologin** program:

/sys/siologin/siologin *siologin_arg_list*

Special Considerations

Use the log files **'node_data/siomonit_log** and **'node_data/siologin_log** to help you debug **siologin** or **siomonit** server problems. Figure 3-13 shows a sample of an **siomonit_log** file.

```
Tuesday, February 19, 1985  9:28:13
     Quit fault. Restarting any dead processes...
Tuesday, February 19, 1985  11:05:56
     ** Process didn't stay alive for 15 secs: **
   /sys/siologin/siologin  /dev/sio1 -n siologin1
Tuesday, February 19, 1985  11:08:15
     ** Received stop fault. Closing up shop. **
Tuesday, February 19, 1985  16:34:53
   Restarted process:   /sys/siologin/siologin /dev/sio1 -n siologin1
Tuesday, February 19, 1985  9:18:02
 * Couldn't open command input file `node_data/siomonit_file. Status 1010015 *
```

Figure 3-13. Sample 'node_data/siomonit_log File

Often, the failure of an **siologin** process to stay alive can be caused by a locked SIO line (**/dev/sio***x*). For example, when a user on a locally connected terminal does not end the session with CTRL/Z, the SIO line to that terminal remains locked. To free up the line, use **sigp -q** to prod **siomonit** into trying again. After you free the line, you may want to include **ulkob ^1 -f** in the appropriate start-up file.

If **siomonit** terminates and its child processes do not also terminate, the child processes are, in effect, orphans. The existence of the orphans interfere with **siomonit**'s attempts to restart new child processes when it restarts. The **siomonit** program itself never stops a child process, its own process, or an orphan process. However, **siomonit** will not be notified when an orphan process terminates. Instead, if **siomonit** detects an orphan's existence, it wakes up every 15 minutes to see if the orphan has stopped. If the orphan process has ended, **siomonit** restarts the **siologin** process as directed in its argument list. Should this occur, a user may have to wait 15 minutes before completing the **siologin** process.

3.13.6 The Clock Server: cron

The **cron** server executes commands stored in the **/usr/lib/crontab**, and **/usr/lib/crontab.local** files. These two files specify the command to be executed and the date and time to execute the command. The **cron** process reads the files once every minute and executes the specified commands at the specified times. See the **cron** manual pages for more information on using **cron**.

You should start **cron** from the **/etc/rc** start-up file to ensure it has root access. Once started, **cron** does not stop until the node is brought down. (Note that like all servers started from **/etc/rc**, you must also create a file named "cron" in the **/etc/daemons** directory to start **cron**. See "Start-Up Procedures" for general information on server start-up.)

See the manual pages for **cron** for details on how to create the **crontab** files read by **cron**.

3.13.7 The Remote User Communication Server: talkd

The **talkd** server is used by the **talk**(1) program. The **talkd** server establishes the connection for remote communications sessions. The **talkd** server maintains the network addresses and other parameters necessary to establish a stream connection through which communications take place. See the **talkd** manual pages for more information on using **talkd**.

You should start **talkd** from the **/etc/rc** start-up file to ensure it has root access. (Note that like all servers started from **/etc/rc**, you must also create a file named "talkd" in the **/etc/daemons** directory to start **talkd**. See "Start-Up Procedures" for general information on server start-up.)

3.13.8 The Server for the Write Program:

The **writed** server is used by the **write**(1) program. **Write** enables allows communication between two nodes by copying lines from one node to another. To enable this type of communication, **writed** must be running on both the sending and receiving nodes.

The **writed** server starts an **mbx_helper** automatically when it is invoked. See the **writed** manual pages for more information on using **writed**.

You should start **writed** from the **/etc/rc** start-up file to ensure it has root access. (Note that like all servers started from **/etc/rc**, you must also create a file named "writed" in the **/etc/daemons** directory to start **writed**. See "Start-Up Procedures" for general information on server start-up.)

3.14 Log-In Monitoring

BSD provides node log-in monitoring for log-in attempts via

- The Display Manager
- A window
- The **spm**
- An SIO line

When you configure login monitoring, you can specify whether records of successful and unsuccessful logins or only unsuccessful logins should be kept. You can also specify the sources of the logins you want to monitor.

3.14.1 Configuring the Log-In Facility

The file **'node_data/login_log.config** specifies which attempts to log in should be recorded and the log file in which to store this information. The default log file is **'node_data/login_log**.

The **'node_data/login_log.config** file contains a line for each type of login that can be monitored. The sample line, shown below, is for logins via the Display Manager:

```
# -display [-inv_only]
```

To enable log-in monitoring uncomment the appropriate lines. If you don't uncomment any lines and if you don't create the log file, no logging will take place. To record only invalid log-in attempts, remove the square brackets from around **-inv_only**.

To specify a log file other than **'node_data/login_log**, uncomment the following line and replace **path_name** with the name of the log file you want.

```
#-file  path_name
```

Note that if the log file you specify does not exist, no logging will take place. You must create an empty log file at the location you specify.

3.14.2 The Log-In Facility Log File

Information about the logins is stored in a log file. The default log file is named **'node_data/login_log**. Note that the log file grows without bounds and must be managed.

For each login, the log file contains:

- The date and time of the login
- An indication of whether the login was from the Display Manager, a window, the SPM, or an SIO line using **siologin**
- The user ID under which the login was initiated
- An indication of whether or not the login was successful (unless you specify that only unsuccessful logins should be recorded)

Figure 3-14 is a sample log file.

```
Friday, October 30, 1987  16:12:45 window  odonnell.osdev.r_d  invalid login attempt
Friday, October 30, 1987  9:30:09 window  jjdjd  invalid login attempt
Friday, October 30, 1987  13:13:33 display  doug.osdev.r_d.5FE  logged in
Friday October 30,  17:36:26 window  root.staff_sr9.none.5FE  logged in
Friday, October 30,  1987  17:42:39 display  doug.osdev.r_d.5FE  logged in
Monday, November 2, 1987  15:09:28 sio  odonnell.osdev.r_d  logged in  connected to device:  /dev/sio1
Monday, November 2, 15:09:50 sio  odonnell.osdev.r_d.5FE  logged out from device:  /dev/sio1
Monday, November 2, 15:15:50 sio  intruder  invalid login attempt  on device: /dev/sio2
```

Figure 3-14. Sample Log-In Monitoring Log File

3.14.3 Log File Protection

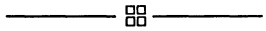
To protect the log file from unauthorized modification, you should give it fairly restricted protection. If you do this, you must also stamp the file as a login protected subsystem data object to allow the login program to open it and make entries. You should also protect the **login_log.config** file appropriately. The following settings are suggested for the log file:

Acl for login_log: subsystem login object

Required entries:

root.%.%	pwrx--
%.sys_admin.%	pwrx--
%.%.none	[ignored]
%.%.%	--r-k-

(Note that in the table, the **k** in the entry for **%.%.%** makes it unnecessary to protect the **'node_data** directory.) In some circumstances, you may want to add an entry that gives the node owner full rights to the file or make the node owner the owner of the file.



Chapter 4

Creating and Maintaining the Registry

Contents

Chapter 4

Creating and Maintaining the Registry

The Domain/OS registry consists of a database of account information and associated software that enables you to store and manipulate account information, thereby controlling access to the computers on your network. The registry database can be replicated so that it resides on several nodes in a network or internet. A server must run on each of these nodes.

This chapter focuses on the **network registry**, a registry of account information that is accessible throughout your network or internet via registry servers. In addition to the network registry, you can create **local registries** residing and operating on individual nodes. Ordinarily, local registries are used only as a backup when the network registry is unavailable. Chapter 4 also provides the following information:

- Overviews of the registry and the registry server
- A discussion of network registry administration and local registries
- A description of how to operate small networks with local registries

Through the rest of this chapter, the term "registry" refers to the network registry (database and software). The term "local registry" always appears in full.

4.1 Registry Software

The registry server manages changes to the data and maintains consistency among the replicas of the registry database. Any node must communicate with a registry server when it requires access to registry information—for instance, when it checks an attempt to log in. Nodes use the Location Broker, a component of the Network Computing System™, to locate registry servers.

Figure 4–1 shows a sample network configuration, including a master registry node, a replica registry node, and normal nodes. The figure indicates where the registry server (**rgyd**) and the Location Broker daemons (**glbd** and **llbd**) are running, where replicas of the registry database reside, and where local registries reside.

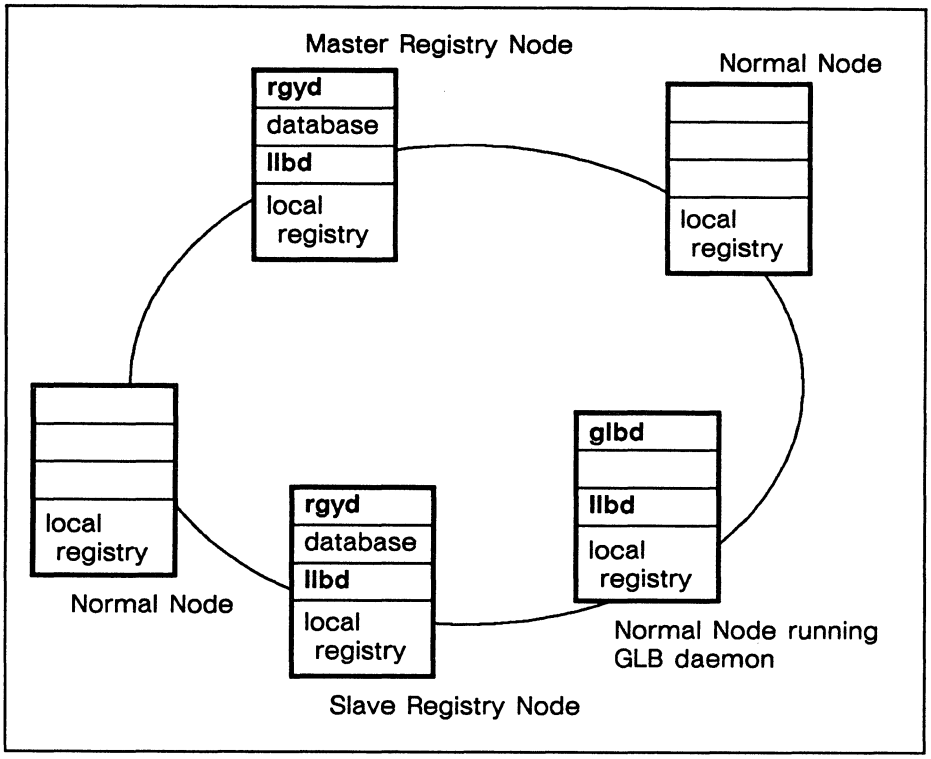


Figure 4–1. Registry Components

4.1.1 The Registry Server

The registry server, **/etc/rgyd**, can run either as a **master** or as a **slave**. There is exactly one master server on a network or internet. All other servers are slaves.

The master server handles operations that change the database (for example, adding an account or changing a password) and propagates the changes to the slaves. Either the master or a slave can handle operations that only read the database (for example, verifying a password when a user logs in). Thus, the master server must be available for any editing operation, but a slave server suffices for lookup operations. Section 4.4 describes how the servers maintain consistency among the databases.

While it runs, the registry server, whether a master or a slave, maintains its database in virtual memory. The server periodically saves the database to a copy that resides on disk. Because the copy on disk is not always up to date, there are special procedures you should follow for backing up the registry database and for restoring the database from a backup tape. These procedures are detailed in Sections 4.8, "Routine Manitenance," and 4.10, "Troubleshooting."

We recommend that, in an internet, at least one registry server exist on every network. We discuss considerations for configuring servers in Subsection 4.6.1.

4.1.2 Tools for Editing and Administering the Registry

Nearly all editing of the registry database must be done with **edrgy**. (The only exceptions are certain operations that users can perform on their own accounts, such as changing passwords.) The master registry server must be available to receive changes.

The **rgy_admin** tool administers the registry server. It can list, reset, replace, and delete replicas of the registry. It also performs special functions such as changing the master registry site and reinitializing a slave registry server.

The **cvtrgy** tool enables you to run SR10 and pre-SR10 registries simultaneously on a network or internet. See *Making the Transition to SR10 Operating System Releases*.

If you are creating a Domain internet, you can use **rgy_merge** to merge data from registries that have operated in partitions of the internet. See *Managing Domain Routing and Domain/OS in an Internet*.

If your Domain systems coexist with other UNIX systems, you should ensure that name and account information is consistent between the Domain/OS registry and the UNIX password and group files. The **import_passwd** tool, described in Section 4.11, helps you to establish consistency.

4.1.3 Location Broker Software

As we mentioned earlier, nodes use the Location Broker to locate registry servers. Two daemons, the Local Location Broker daemon (**llbd**) and the Global Location Broker daemon (**glbd**), are essential to Location Broker operation and hence to operation of the registry. A **glbd** must run on at least one node in a network. In an internet, each network must have at least one **glbd**. An **llbd** must run on all registry server nodes and on all **glbd** nodes. *Managing the NCS Location Broker* provides information about starting and running these daemons.

4.2 The Registry Database

The registry database stores information about **names**, **taccounts**, and **policies**. Table 4-1 shows the general categories of data in the registry database.

Table 4-1. Registry Database Categories

Names	Accounts	Policies
Persons	SIDs (pgo triplets)	Registry Policy
Groups (member lists)	Passwords	Organization Policy
Organizations (member lists)	Home Directories	
	Log-In Shells	

All of these registry database objects can be edited with the **edrgy** command.

4.2.1 Names

Names establish the existence of individual persons, groups, and organizations within the registry. Identical names can exist in different domains of the registry database—there can be a person, a group, and an organization with the same name.

Persons can have a **primary name** and one or more **aliases**; an alias is an alternate name for a person. Groups and organization have only primary names, not aliases.

Associated with each primary name are a **UID (Unique Identifier)** and a **UNIX number** that the operating system uses as identifiers. A primary name or alias can also have a **fullname**. The operating system doesn't use fullnames but provides them for easy recognition by users.

Unique Identifiers

The operating system associates each primary name with a (UID) consisting of a node ID and a time stamp. When printed as a text string, a UID has two parts separated by a period, as in **38edde17.50007c5f**.

The association between UID and primary name is fundamental to the protection of files and directories in your network. If you destroy this association (for example, by deleting the name from the database), you effectively "orphan" all files owned by that UID. You can use **edrgy** to "adopt" the orphaned files (that is, associate the UID with a name).

Numbers

A **number**, in the context of the registry database, is a decimal identifier associated with a primary name. Numbers are analogous to UNIX user and group IDs, so we sometimes refer to them as **UNIX numbers** or **UNIX IDs**. They exist for UNIX compatibility and are used mainly by UNIX programs.

If you convert a registry from SR9 to SR10, the conversion tool adds numbers to the registry database; if you are building a new registry database, you assign numbers when you create names. If your Apollo systems share files with other UNIX systems, you should ensure that names, UNIX IDs, and account information are consistent between the Domain/OS registry and the foreign systems. Subsection 4.6.5 contains more information about establishing uniform UNIX IDs.

Aliases

Aliases enable you to establish several accounts that have different home directories and passwords but share the same access rights to files. You can use **edrgy** to change an alias into a primary name and a primary name into an alias. Although it is possible to create an alias without an existing primary name, you ordinarily create a primary name entry before you create any aliases for that name. Groups and organizations can have only primary names, not aliases.

Fullnames

A **fullname** is a text string associated with a person, group, or organization name. The fullname typically describes or expands the name. For example, the person name **owright** could have the fullname **Orville Wright**, and the organization name **rd** could have the fullname **Research and Development.**

4.2.2 Accounts and Subject Identifiers

Accounts define who can log in to the computers on your network. An account associates a person, a group, and an organization. Each account includes information that the operating system needs when a user logs in, such as a password and a home directory.

Subject Identifiers

The operating system identifies each account by a **Subject Identifier** (**SID**) consisting of a person name, a group name, and an organization name, separated by periods, as in **rubinstein.pianists.none**. We use the terms **pgo** and **pgo triplet** as synonyms for SID.

In some SIDs, a percent character (%) can appear as a wildcard in place of a name. For instance, **mozart.%.%** matches **mozart.pianists.none** and **mozart.symphonists.classical**. Wildcards are not allowed in account SIDs, but they are allowed in the SIDs that specify ownership of registry data.

Other Account Information

For each account, the operating systems stores the following information in the registry database:

- An SID, for example, **grappelli.violinists.jazz**.
- An abbreviation for the SID, for example, **grappelli** or **grappelli.violinists**. At login, a user need only enter the abbreviation to specify the SID. When you add an account, **edrgy** automatically assigns the shortest unique abbreviation unless you specify otherwise. For example, if the SID **babar.elephants.none** already exists with the abbreviation **babar**, a new account for **babar.kings.none** will have the abbreviation **babar.kings**.
- An encrypted password. When a user attempts to log in, the operating system prompts for the password, encrypts the text string that the user enters, and checks the result against the encrypted string stored in the database.
- A flag that determines whether the account is valid. If an account is not valid, the operating system will reject any attempt to log in on that account.
- A home directory.
- A log–in shell.
- Miscellaneous information. This is a text string that typically describes the user of the account. If the person specified in the SID has a fullname, the person's fullname is concatenated with the account's miscellaneous information to form the *gecos* field in the **/etc/passwd** file.

4.2.3 Reserved Names and Accounts

We supply several **reserved** names and accounts with the operating system, for use by various system operations. You cannot change the UIDs and numbers associated with reserved names.

Table 4–2 lists these names and accounts.

Table 4–2. Names, Accounts, and SIDs

Person	Group	Organization	Person.Group.Organization
admin bin daemon lp root sys_person user uucp none	backup bin daemon locksmith login mail none server sys staff sys_admin sys_proj wheel	apollo none sys_org	none.none.none user.none.none sys_person.none.none admin.none.none daemon.none.none bin.bin.none lp.bin.none uucp.daemon.none root.staff.none

In addition to those required for reserved accounts, we supply the following memberships: the person **user** is member of the groups **backup** and **sys_admin** and the organization **apollo**; the person **bin** is a member of the group **mail**, and the person **root** is a member of the groups **bin** and **sys**.

4.2.4 Groups and Organizations

Here we describe some features that are specific to groups and organizations.

Passwords

Each group or organization can have a password. In the SysV environment, if a group has a password, a user who is not a member of the group can acquire its privileges by invoking the **newgrp** command and entering the correct password.

Many sites opt not to assign passwords for groups and organizations.

Membership Lists

Each group or organization has a **membership list** containing the person names of all its members. In order for you to create an account, the specified group and organization must contain the specified person in their membership lists.

For example, in order for you to create an account **mahler.symphonists.none**, the group **symphonists** and the organization **none** must have the person **mahler** in their membership lists. (If you are the owner of the group **symphonists** and the organization **none**, **edrgy** automatically adds **mahler** to their membership lists so that you can create the account in one step. If you are not the owner of **symphonists** and **none** and **mahler** is not already on these lists, **edrgy** issues an error. We describe owners of registry objects in Subsection 4.2.7.)

Project Lists

BSD UNIX systems associate each user process with not one group but a set of groups. This set consists of the group specified for the user in **/etc/passwd** and any other groups of

which the user is listed as a member in **/etc/group**. The user always has the access rights that accrue from membership in every group in the set.

Domain/OS creates such a set of groups, called a **project list**, if you have the PROJLIST environment variable set to **true**. The project list consists of all groups of which the user is a member. If PROJLIST is not set, only the **pgo** of the user process controls access to files. Thus, Domain/OS allows you to choose either BSD or SysV behavior.

By default, PROJLIST is set automatically if your SYSTYPE is **bsd4.3**.

Not every group can be included in a project list. The reserved group **locksmith**, for instance, has a property that prohibits its inclusion in project lists. When you create a group with the **edrgy** command, the group is set up to allow inclusion in project lists unless you explicitly specify otherwise. If you change the properties of a group to disallow its inclusion in project lists, you cannot undo the change.

4.2.5 Policies

The registry database includes **policies** that regulate the following aspects of accounts and passwords:

- Password expiration date
- Password lifespan (how long a password remains valid before it must be changed)
- Account lifespan (how long an account remains valid)
- Minimum length of passwords
- Whether a password can consist entirely of spaces
- Whether a password can consist entirely of alphanumeric characters

You can use **edrgy** to set any of these policies for the registry as a whole or for a particular organization. If a policy is set both for the registry and for an organization, the stricter policy applies. For example, if registry policy specifies a minimum password length of six characters and policy for the **rd** organization specifies eight characters, the account **bell.comm.rd** must have a password at least eight characters long.

By default, when the registry is created, policies are at their most permissive—passwords have no expiration date, passwords and account have unlimited lifespans, there is no minimum password length, and passwords can consist entirely of spaces or entirely of alphanumeric characters. Many sites set one or more of these policies to be stricter.

4.2.6 Properties

The registry database also includes the following **properties:**

- The owner of the registry as a whole
- The owner of each name domain
- Whether UNIX restrictions are enforced
- Whether UNIX restrictions are met
- Whether the registry database is read-only

You can use **edrgy** to set any of these properties.

UNIX restrictions are met if no names exceed eight characters, all accounts have the **p** (person only) abbreviation, and all passwords are valid for vanilla UNIX systems. If UNIX restrictions are enforced, all data in the registry database must meet the restrictions. (Reserved names and accounts are excepted; they do not affect whether UNIX restrictions are met and they are not affected if the restrictions are enforced.)

4.2.7 Owners

Every object in the registry database has an owner who is allowed to modify that object. There are also owners for the registry as a whole and for each of the three name domains (person, group, and organization).

Owners are represented by SIDs. These SIDs can include wildcards.

Default Ownerships

All ownership information is contained in the registry database and can be edited via **edrgy**. When the registry is first created, the initial data, the name domains, and the registry as a whole are all owned by **%.%.%**. By default, all data added via **edrgy** is owned by **%.sys_admin.%**.

Some sites assign owners other than the defaults for registry information. You can use the **change** command in **edrgy** to assign ownerships for any names. You can use the **prop** command to assign ownerships for the three name domains and for the registry as a whole. To specify the default owner for data added via **edrgy**, issue the **defaults** command at the beginning of every **edrgy** session.

At most sites, it is simplest to have one owner for the registry and all of its data. This owner can be the unrestrictive SID **%.%.%**, a restrictive SID such as **%.sys_admin.%**, or an SID created specially for registry administration such as **%.registry_admin.%**. However, some sites prefer to have different owners for different data. A common scheme is to use organizations as administrative domains. For example, you can assign **%.sales_admin.sales** as the owner for the **sales** organization; this SID will have control over the membership list for **sales** and thereby will have control over all accounts of the form **%.%.sales**.

Rights of Owners

The following list summarizes the operations that can be performed by the owner of the registry and by owners of registry objects:

- The owner of the registry can change registry policies and properties, change the owners of name domains, and use the **rgy_admin** and **rgy_merge** tools.

- The owner of a name domain can add entries in that domain. For instance, the owner of the person domain can add persons.

- The owner of an organization can add or delete members, change properties such as the password and the fullname, change policies for that particular organization, or delete the organization. If the owner of *organization* deletes *person* from the membership list, any accounts for *person.%.organization* are also deleted. If the owner deletes *organization* itself, accounts for *%.%.organization* are deleted.

- The owner of a group can add or delete members, change properties such as the password and the fullname, or delete the group. If the owner of *group* deletes *person* from the membership list, accounts for *person.group.%* are also deleted. If the owner deletes *group* itself, accounts for *%.group.%* are deleted.

- The owner of a person can change properties such as the fullname, delete the person, or add an account (provided the person is a member of the specified group and organization). Deleting *person* also deletes accounts for *person.%.%*.

- Account information such as the password and fullname can be changed by any user of the account.

- The owner of any registry object can change the owner of that object. (An owner can actually "give away" ownership.) This also applies to ownership of the registry as a whole.

Any of these registry operations can also be performed by a **root.%.%** or **%.locksmith.%** account, provided the user is logged in at the master registry node.

4.3 The /etc/passwd, /etc/group, and /etc/org Files

UNIX systems store information about accounts in the files **/etc/passwd** and **/etc/group**. Domain/OS provides these files and, in addition, an **/etc/org** for information about organizations.

In Domain/OS, the **passwd**, **group**, and **org** files cannot be directly edited. Changes to the registry database are made with **edrgy** or with specialized utilities such as **/com/chpass** and **/bin/passwd**. The registry server reflects all changes in the **passwd**, **group**, and **org** files as well as in the registry database. The server updates these files periodically, whenever it saves its database from memory to disk.

Each registry server, whether master or slave, maintains its own versions of the **passwd**, **group**, and **org** files in its copy of the registry database. On all nodes, the files in **/etc** are specially typed file system objects; when a user requests access to one of the files, the operating system finds a version of the file at one of the registry server nodes.

UNIX system calls such as **getpwent**(3) access the version of the registry database that resides in memory at a registry server node. These calls do not use the **passwd**, **group**, and **org** files at all. Thus, though the files in **/etc** are not always current with the registry database in memory, the asynchrony is harmless.

The rest of this section describes the format of the **passwd**, **group**, and **org** files in Domain/OS. See also the reference documentation for **passwd**(5), **group**(5), and **org**(5).

4.3.1 The /etc/passwd File

The **passwd** file contains one line of text for each account in the registry database. Colons divide each line into seven fields, as follows:

account-name:*encrypted-password*:*user-ID*:*group-ID*:*gecos*:*home-directory*:*login-shell*

The *account-name* is the SID of the account, abbreviated if possible. The *user-ID* and *group-ID* are the UNIX numbers of the person and group in the account SID. The *gecos* field is the concatenation of the person's fullname and the account's miscellaneous information. (Subsection 4.2.2 explains the SID, miscellaneous information, and other account data.)

The **passwd** file can include several entries for one person if the person has several accounts, as in the following excerpt:

```
arnold:QB4mmrONjs/szD:15586:98:Ken Arnold://anarres/arnold:/bin/csh
arnold.unix:ZcdWqJGsemkYJn:15586:38:Ken Arnold://anarres/arnold:/bin/csh
```

If an account is invalid, its entry in the **passwd** file will contain the text string ***NOACCT*** where the encrypted password would ordinarily appear.

4.3.2 The /etc/group File

The **group** file contains one line of text for each group in the registry database. Colons divide each line into four fields, as follows:

group-name:*encrypted-password*:*group-ID*:*membership-list*

The *group-ID* is the UNIX number of the group. The fourth field is a list of person names, separated by commas. Thus, part of a **group** file could look like this:

```
dds::78:emartin,pato,phl,pjl,mishkin,molson,mk
debug::211:gordon,cas,gkk
demo::68:
```

In the **group** file, each entry occupies one line of text.

4.3.3 The /etc/org File

The **org** file contains one line of text for each organization in the registry database. Its format is the same as that of the **group** file:

org-name:*encrypted-password*:*org-ID*:*membership-list*

The *org-ID* is the UNIX number of the org. Part of an **org** file could look like this:

```
intl::9:jimk
legal::10:barker_c,olesen_d,ponte_l,fazio_p,rossini_r,peter,\
    lewis_j,dacier_p,lynch_p,egoldman,barbara
```

In the **org** file, a backslash (\) at the end of a line indicates that the membership list continues on the next line.

4.4 How the Registry Database is Replicated

As we mentioned in Section 4.1, the registry database can be replicated. In addition to the database managed by the master server, you can create database replicas managed by slave servers. Replication of the registry database can enhance performance and reliability, especially in an internet.

The registry uses "weakly consistent" data replication: the data in every replica of the database are always available, though the data in slave databases are not always up to date. The replication mechanism ensures that all replicas of the database will converge to the same set of information while remaining available for lookup by system software and by applications.

Figure 4–2 shows part of a replicated registry configuration. The master server and one of the slave servers are pictured. Each server manages a database of name and account information and a **replica list** containing the network address of each host that runs a registry server. The master server also manages a **propagation queue** containing information that it must propagate to the slaves.

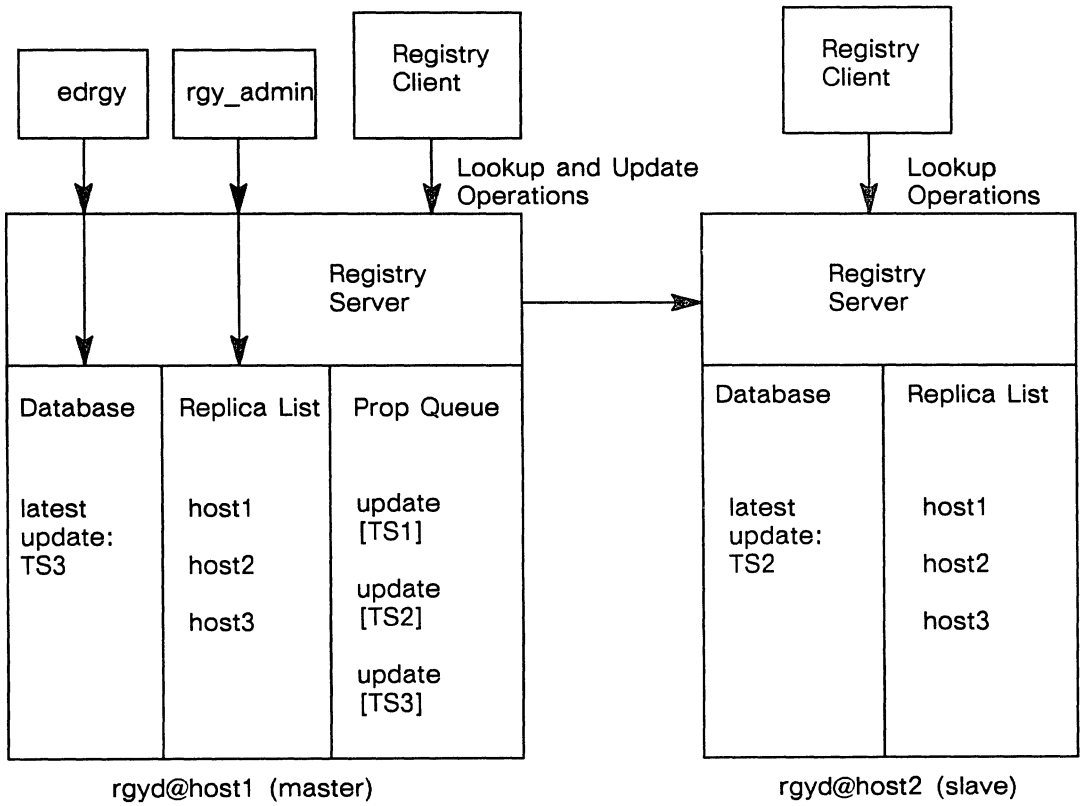


Figure 4–2. Registry Server Operation

As the figure shows, **edrgy** makes changes only to the database, and **rgy_admin** makes changes only to the replica list; these changes can be made only through the master server.

When you change any information in the database or in the replica list (for example, by adding an account, or adding a replica) the master server enters the changed information, called an **update**, in its copy of the database. It marks both the database and the update with a time stamp.

The master server places the update in the propagation queue and attempts to propagate the update to each slave server on its replica list. If propagation of an update to one of the slaves does not succeed on the first attempt, the master server retries periodically until it succeeds. The master server always propagates updates in chronological order, according to their time stamps. It keeps track of which updates are pending propagation to which slaves. When an update has propagated to all the slaves, it is removed from the propagation queue.

In Figure 4–2, for example, the update with time stamp **TS3** awaits propagation to the slave server at **host2**. Propagation of the updates with time stamps **TS1**, **TS2**, and **TS3** to the slave server at **host3**, not shown, is still pending.

If the master server loses communication with a slave server for a long time, the master server will reinitialize the slave server, giving it a fresh copy of the entire database, when communication is restored.

4.5 How the Registry Database is Stored on Disk

Each registry server, whether master or slave, maintains a working copy of its database in virtual memory and a permanent copy on disk. All lookups and updates operate on the copy in virtual memory. The server uses the copy on disk to initialize the copy in memory when it starts up.

When a master server receives an update from a utility such as **edrgy** or when a slave server receives an update from the master, the server applies the update to its database in virtual memory and also saves the update in a log file on disk. Updates accumulate in this log file in chronological order. When a server restarts (for example, when a server node boots), it initializes its database in memory from the database on disk and then it "replays" the updates from the log file. This mechanism ensures that no updates are lost when a server node is shut down.

Each registry server periodically saves its entire database from virtual memory to disk. The database is stored in the directory **/sys/registry**. When it performs a disk save, the server clears the log file.

Sections 4.8, "Routine Maintenance" and 4.10, "Troubleshooting," give procedures for backing up the registry database and for restoring the database from a backup tape.

4.6 Setting Up the Registry

This section describes how to set up the registry. Note that several of the procedures in this section must be performed by the **root** user.

In Subsection 4.6.3, you must choose one of two procedures, depending on whether you are creating a new network of nodes running SR10 or updating an existing network from SR9 to SR10.

The procedures in Subsections 4.6.5 and 4.6.8 might not be necessary at your site. All other procedures in this section are essential to successful operation of the registry.

If you are creating a Domain internet, you should see *Managing Domain Routing and Domain/OS in an Internet* for more information.

4.6.1 Planning a Configuration

Use the following considerations to plan a configuration of registry servers for your site:

- You can run more than one **rgyd** on a network. In an internet, we recommend that you run at least one **rgyd** in every network.
- Nodes that run **rgyd** should have local disks. They also should have at least 3 MB of physical memory, more for very large registries.
- Nodes that run **rgyd** should be running and available all the time. It is especially important that the node where the master server runs be available throughout the network or internet.

Choose a site for the master server. If you decide to run several servers, choose sites for the slave servers.

4.6.2 Starting Location Brokers

Before you run registry servers, you must establish Location Brokers on your network. At least one Global Location Broker (**glbd**) must run on your network. In an internet, at least one **glbd** must run on each network of the internet. A Local Location Broker (**llbd**) must run on every node that runs a **glbd** and on every node that will run a registry server (master or slave).

Managing the NCS Location Broker gives procedures for starting up Location Brokers. Refer to this manual now and then continue with Subsection 4.6.3.

4.6.3 Creating the Registry Database

There are two ways to create an SR10 registry database.

Updating an SR9 Registry to SR10

If you are updating from SR9 to SR10 system software on your network or internet, you should convert your SR9 registry database to SR10 format via the **cvtrgy** utility. Skip the rest of this subsection, follow the conversion procedures described in *Making the Transition*, then proceed with Subsection 4.6.4.

Setting Up a New SR10 Registry

If you are setting up a new network or internet of Domain nodes and SR10 Domain/OS is the only system software any of these nodes have ever run, you use the **rgy_create** utility to create a new database and initialize it with reserved names and accounts. You typically create the SR10 registry as part of your first SR10 installation. Since it is used only once, **rgy_create** resides in the **/install/tools** directory and is not included in subsequent SR10 installations. See *Installing Software with Apollo's Release and Installation Tools* for more information about **rgy_create**.

4.6.4 Starting the Master Registry Server

Log in on the master registry site node and use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node.

To start the master server, complete the following steps:

1. Become **root** and enter the following command on the master node:

   % /etc/server -p /etc/rgyd &

   The **-p** option causes **/etc/server** to create a process that runs under the SID that invoked the command, rather than the default of **user.server.none**.

2. Create the file **/etc/daemons/rgyd** on that node, so that the server will start each time the system boots. Use the UNIX **touch** command or the Aegis **crf** command:

   % touch /etc/daemons/rgyd (UNIX environments)

   $ crf /etc/daemons/rgyd (Aegis)

4.6.5 Establishing Uniform UNIX Numbers

If your Apollo systems share files with other UNIX systems, you should ensure that names, UNIX IDs, and account information are consistent between the Domain/OS registry and the foreign **passwd** and **group** files. For the purposes of this discussion, "file sharing" can take the form of direct access through facilities such as Domain NFS or indirect file transfer via media such as **tar** tapes.

We provide a tool called **import_passwd** that helps you to identify and resolve conflicts of names, UNIX IDs and account information. If you plan to share files between Apollo systems and other UNIX systems, we recommend that you run **import_passwd** now to minimize the number of changes you have to make. Typically, you run **import_passwd** in a mode that changes IDs in the Apollo registry to match the IDs on the foreign systems; afterward, you will have to run another tool called **syncids** to ensure that the IDs stored in Apollo file systems match those stored in the registry.

See Section 4.11 for detailed information about **import_passwd**.

4.6.6 Setting Policies, Properties, and Passwords

Before you make any further changes to the registry database, you should use the **prop** command in **edrgy** to view policies and properties and change them as desired.

We provide default passwords for all reserved accounts except **user.none.none**, which by default has no password. These defaults are set either by **cvtrgy** (if you converted from an SR9 registry) or by **rgy_create** (if you created a new SR10 registry). You should use **edrgy** to change or set the passwords for these accounts.

Now is also the best time to change the owners of reserved registry data, if you do not want to use the defaults. See Subsection 4.2.7 for details.

4.6.7 Adding Names and Accounts

Your registry now contains the following names and accounts:

- Those added as reserved information by **rgy_create** or **cvtrgy**
- If you used **cvtrgy**, those derived from your SR9 registry
- If you used **import_passwd**, those derived from the password and group files on your other UNIX systems

Use **edrgy** to add any other names and accounts that your site requires. You can do this now or at any time later.

4.6.8 Creating Slave Registry Replicas

Follow the procedures in this subsection if you want to replicate the registry database.

1. Log in on a slave node and use **/bin/ps** or **/com/pst** to verify that an **llbd** is running on the node.
2. To start the slave server, become **root** and enter the following command on the slave node:

   % /etc/server -p /etc/rgyd -create &

   This command locates the master server, adds the local node to the master replica list, and causes the master to initialize a new database at the local node.
3. As you did on the master node, create the file **/etc/daemons/rgyd**, so that the server will start each time the system boots. Use **touch** or **crf**:

   % touch /etc/daemons/rgyd (UNIX environments)

   $ crf /etc/daemons/rgyd (Aegis)
4. Repeat the above steps for each replica you want to create.

   You can use the **lrep -state** command in the **rgy_admin** tool to verify that the master and slave servers are running.

4.6.9 Restarting Registry Servers

To restart a registry server, whether a master or a slave, run **rgyd** without any options:

```
% /etc/server -p /etc/rgyd&
```

4.7 Managing the Registry

Managing the registry requires you to administer both the information in the registry database and the operation of the registry server. You use **rgy_admin** to administer the operation of the registry server and **edrgy** to edit data in the registry database.

4.7.1 Managing the Registry Server

The **rgy_admin** tool manages the operation and replication of the registry server. You can use **rgy_admin** to read and edit replica lists, delete replica registries, and stop registry servers.

When invoked, **rgy_admin** issues a prompt, **rgy_admin:**, and enters an interactive mode, waiting for your input. Sections 4.8, 4.9 and 4.10 contain procedures for some tasks you might need to perform with **rgy_admin**.

4.7.2 Editing the Network Registry Database

The **edrgy** command manages the name and account information in the registry database. Using **edrgy**, you can add, edit, or delete any of the following information:

- Information about persons, groups, and organizations
- Group and organization membership lists
- Policies for organizations and for the registry as a whole
- Properties of the registry
- Account information
- Owner information

You can also use **edrgy** to view registry database information without making changes.

4.7.3 Editing the Local Registry Database

You can use the **-l** option of **edrgy** to edit a node's local registry information. This form of the command is available to ordinary users and not reserved to the registry owner. Remember, however, that at each login, the operating system updates any entry in the local registry for the account used.

4.7.4 Merging Disjoint Registries

Network reconfigurations sometimes require you to merge the databases of two registries that have been operating independently.

We provide the **rgy_merge** tool to help you merge registry databases. This tool compares two master registry databases and informs you if any names or numbers conflict. You must use **edrgy** to resolve any conflicts before **rgy_merge** will actually perform the merge.

There are two types of reconfigurations that require merging of registries:

- Connecting two networks to create an internet
- Combining two networks into one larger network

Note that in this context, a "network" can consist either of several nodes or of a single standalone node running a network registry.

If you are creating a Domain internet, see *Managing Domain Routing and Domain/OS in an Internet*. This book describes all aspects of creating an internet and includes procedures to merge registries.

If you are combining two networks into one larger network, the procedures for merging registries in *Managing Domain Routing and Domain/OS in an Internet* still apply. You should perform the merge during off hours with as few users on the networks as possible. Until the merge is complete, two disjoint registries will exist on one network, so the chances of obtaining incorrect information in a registry lookup are greater than in the internet case.

Section 4.9 contains several procedures for dealing with network reconfigurations.

4.8 Routine Maintenance Procedures

This section contains procedures for routine maintenance of the registry.

4.8.1 Backing Up the Registry Database

Because the registry server maintains its most current data in memory and saves data to disk only periodically, we provide a special procedure for backing up the database. You should follow Procedure 4–1 either when you back up the disk containing the master replica of the database or when you back up the database only. The server will not accept updates while the backup is in progress.

Procedure 4–1. *Backing Up the Registry Database*

You must be an owner of the registry to perform Tasks 1 and 3 of this procedure.

Task 1: Put the master server in maintenance state

Use the **state** command in **rgy_admin** to put the master server in maintenance state. While in maintenance state, the server will refuse updates.

Task 2: Back up the master replica

Back up the master replica of the registry database by backing up either the entire volume or the **/sys/registry** tree.

Task 3: Take the master server out of maintenace state

Use the **state** command in **rgy_admin** to take the master server out of maintenance state. The server will resume accepting updates.

4.8.2 Checking Consistency of Registry Replicas

You can use Procedure 4–2 to check that all replicas of the registry are operating and that all copies of the database are up to date.

Procedure 4–2. *Checking Consistency of Registry Replicas*

Task 1: Invoke rgy_admin

Task 2: Use the lrep –state command

The **lrep –state** command reads the replica list at the **rgy_admin** default host, queries each replica in the list, and displays information about the each replica that responds. The output of **lrep –state** shows which server is the master, the state of each server, and the time stamp of the most recent update at each replica of the database.

In the following example, **rgy_admin** queries the slave registry server at **//tosca**, which provides information about the master replica at **//aida** and about slave replicas at **//tosca** and **//lulu**. All replicas are operating normally ("in service") and both slaves are in synch with the master.

```
% rgy_admin
Default object: rgy  default host: dds://tosca
State: in service  slave
rgy_admin: lrep -state
dds://tosca          state: in service          1988/04/25.10:49:23
dds://lulu           state: in service          1988/04/25.10:49:23
(master)dds://aida   state: in service          1988/04/25.10:49:23
```

The online and printed reference documentation for **rgy_admin** lists and describes other states that **lrep** may report.

4.8.3 Restarting Registry Servers

If you created the file **/etc/daemons/rgyd** at every registry site, as described in Subsection 4.6.8, a registry server should automatically restart whenever the node boots. In the event that a server fails to restart or dies, you can use Procedure 4–3 to restart it by hand. This procedure applies to either master or slave servers.

Procedure 4–3. *Restarting a Registry Server*

Task 1: Become root

Become **root** on the node where you want to restart the server. Issue the following shell command:

```
% /etc/server -p /etc/rgyd &
```

Task 2: Verify that the server is running

Use the **lrep -state** command in **rgy_admin**, as described in Procedure 4–2, to verify that the server is now running. If it is not, see the troubleshooting procedures in Section 4.10.

Task 3: Verify that /etc/daemons/rgyd exists

Check that the file **/etc/daemons/rgyd** exists so that the server will automatically restart when the system boots.

4.9 Reconfiguration Procedures

This section describes procedures to help you deal with hardware and network reconfigurations.

4.9.1 Changing the Network Address of a Registry Site

If you reconfigure a network or relocate a node that runs a registry server, the network address of a registry site may change. Since registry replicas use their network addresses to locate and identify each other, it is crucial that every replica learn of any changes to network addresses.

If the network address of a registry site has changed, you must stop the registry server at that site (via the **stop** command in **rgy_admin**) and then restart the server (as described in Procedure 4–3). The registry server automatically learns its new network address when it restarts.

Changes of network addresses can be propagated to registry replicas in three ways:

- If only the master replica changes its network address, and all slaves retain their addresses, the master will automatically propagate its new address to all slaves.
- If the master replica does not change its network address, but one or more slaves do, each slave will locate the master and give the master its new address. The master will then propagate the new address to all other slaves.
- If the master replica and one or more slaves change their network addresses, there will be no way for the master and the changed slaves to contact each other, and you should execute Procedure 4–4 to update the replica list at the master.

Procedure 4–4. *Updating the Replica List at the Master Registry*

You must be an owner of the registry in order to perform Task 3 of this procedure.

Task 1: Set the default host to the master registry site

Invoke **rgy_admin**. Set the default host to the master registry site by issuing the **set -m** command.

Task 2: List the addresses of all registry replicas

Use the **lrep -na** command in **rgy_admin** to list the network addresses of all registry replicas (as stored in the replica list at the master site).

Task 3: Update the replica addresses

For any slaves whose network addresses have changed, use the **reprep** command in **rgy_admin** to update these addresses. Use **lrep -na** again to verify that the replica list at the master site is now correct.

Once the replica list at the master site is updated, the master server will propagate changes of network address to the slave servers as necessary.

4.9.2 Changing the Master Registry Site

Smooth operation of the registry requires that the node running the master registry server always be available. If you are planning to remove this node from your network or to shut it down for an extended period, you should change the master registry site.

Procedure 4–5 describes a clean, graceful method for changing the master registry site by causing the master site and a slave site to reverse roles. It assumes that the registry servers at these two sites are operating normally. After the original master has been changed to a slave, you can use Procedure 4–6 to delete it.

Section 4.10 contains procedures for dealing with abnormal situations arising, for instance, from hardware or network failure.

Procedure 4–5. *Changing the Master Registry Site*

You must be an owner of the registry in order to perform Task 3 of this procedure.

Task 1: Choose the master registry site

Choose the new master site. A slave replica must exist at this site. If necessary, create the slave replica, as described in Subsection 4.6.8.

Task 2: Set the default host to the master registry site

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set -m** command.

Task 3: Change the master registry site

Change the master site by issuing the following command:

`rgy_admin:` **change_master -to** *//newmaster*

where *//newmaster* is the new master site you chose in Task 1.

Task 4: Verify that the master site has been changed

Use **lrep -state** to verify that the master site has been changed.

The former master site is now a slave site. You can delete this slave replica by executing Procedure 4–6.

4.9.3 Deleting a Slave Registry Replica

If you are planning to remove a slave site node from your network or to shut it down for an extended period, you should delete the registry replica at that site by following Procedure 4–6.

Procedure 4–6. *Deleting a Slave Registry Replica*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Set the host to the current master registry site

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set -m** command.

Task 2: Delete the slave replica

Delete the slave replica by issuing the following **rgy_admin** command:

`rgy_admin:` **delrep** *//slave*

where *//slave* is the site of the slave replica to be deleted. The master server will instruct the slave replica to delete itself.

Task 3: Verify that the slave has been deleted

Use **lrep -state** to verify that the slave replica has been deleted. The deletion may take a while. Until the slave is actually deleted, **lrep -state** will show the slave as marked for deletion.

4.10 Troubleshooting Procedures

This section contains procedures for troubleshooting the registry. You should not need to use any of these procedures except when network or hardware failures have disrupted operation of the registry.

4.10.1 Recreating a Slave Registry Replica

If a slave replica of the registry database has become corrupted or irrecoverably out of date, you can use Procedure 4–7 to recreate the replica.

Procedure 4–7. *Restoring a Slave Database and Restarting Its Server*

Task 1: Become root

Become **root** on the node where you want to recreate the slave replica. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: Recreate the slave replica

Recreate the slave replica by issuing the following shell command:

```
% /etc/server -p /etc/rgyd -recreate &
```

This destroys the existing database, creates a new one, and starts a slave server.

Task 3: Verify that /etc/daemons/rgyd exists

Check that the file **/etc/daemons/rgyd** exists, so that the server will automatically restart when the system boots.

4.10.2 Recovering from Loss of the Master Registry Replica

If the master registry replica becomes inoperable or unavailable for an extended time, you may need to replace it or restore it. This subsection contains several procedures for recovery of the master replica. You should read through all of them and select the one most appropriate for your particular situation.

If there exists a slave replica with a nearly current database, you can use Procedure 4–8 to turn this replica into the master. (You can check the time stamps of slave databases via the **lrep –state** command in **rgy_admin**.) If no such slave exists, you can use Procedure 4–10 or Procedure 4–11 to restore the registry database from a backup tape.

Procedure 4–8. *Turning a Slave into a Master*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Choose the slave replica

Choose the slave replica that will become the new master.

Task 2: Make the chosen slave the master registry site

Invoke **rgy_admin**. Issue the following **rgy_admin** commands to set the chosen slave site as the default host and to turn this slave into a master:

```
rgy_admin: set -h //newmaster
rgy_admin: become -master
```

where *//newmaster* is the site of the slave replica you chose in Task 1.

Task 3: Verify the new master site

Use **lrep –state** to verify that *//newmaster* is now the master registry site.

If the old master replica becomes available again, you must immediately change either the old master or *//newmaster* to a slave replica, to prevent inconsistencies in the database replicas. Retain the master with the more current database and use Procedure 4–9 to turn the other master into a slave.

Procedure 4–9. *Turning a Master into a Slave*

Use this procedure to turn a master replica into a slave. You should use this procedure only if you have more than one master running on your network or internet, a highly unusual condition. (If you are performing a merge of two disjoint registries, two masters will coexist temporarily, but **rgy_merge** will turn one of them into a slave.)

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Choose the master replica

Choose the master replica that will become a slave.

Task 2: Make the master into a slave replica

Invoke **rgy_admin**. Issue the following **rgy_admin** commands to set the chosen master site as the default host and to turn this master into a slave:

```
rgy_admin: set -h //newslave
rgy_admin: become -slave
```

where *//newslave* is the site of the master replica you chose in Task 1.

Task 3: Verify that the master is a slave

Use **lrep -state** to verify that *//newslave* is now a slave registry site.

Procedure 4–10. *Restoring a Master Server at Its Original Site*

Use this procedure to restore a master replica at its original location. See Procedure 4–11 if you are restoring the database at a different location.

Task 1: Become root and stop rgyd

Become **root** on the master site node. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: Restore /sys/registry

Restore the **/sys/registry** tree from the backup media.

Task 3: Restart rgyd

Restart the server by invoking **rgyd** without any options:

```
% /etc/server -p /etc/rgyd &
```

Task 4: Verify that /etc/daemons/rgyd exists

Check that the file **/etc/daemons/rgyd** exists, so that the server will automatically restart when the system boots.

Procedure 4–11. *Restoring a Master Server at a New Site*

Use this procedure to restore a master replica at a new location. You must be an owner of the registry in order to perform Task 4 of this procedure.

Task 1: Become root and stop rgyd

Become **root** on the new master site node. Use the UNIX **ps** command or the Aegis **pst** command to check whether a **rgyd** is running at the new site. If there is one, stop it via the **stop** command in **rgy_admin**. (You must be an owner of the registry in order to use the **stop** command.)

Task 2: Restore /sys/registry

Restore the **/sys/registry** tree from the backup media. (Alternatively, if the disk that contained the master replica on the original site is still intact, you can move the disk to the new node; in this case, you must run the **chuvol** utility on the volume you are moving.)

Task 3: Restart rgyd

Restart the server by invoking **rgyd** with the **–restore_master** option:

```
% /etc/server -p /etc/rgyd -restore_master &
```

Task 4: Delete the old master site

Use the **delrep –force** command in **rgy_admin** to delete the old master site from the replica list, as described in Procedure 4–12.

Task 5: Create /etc/daemons/rgyd

Create the file **/etc/daemons/rgyd** at the new site, so that the server will automatically restart when the system boots.

4.10.3 Forcibly Deleting a Registry Replica

Use Procedure 4–12 to delete a registry replica when the ordinary method described in Procedure 4–6 has failed.

Procedure 4–12 uses the drastic **delrep -force** command in **rgy_admin** to delete the registry replica from the replica lists at all other registry sites. It does not actually stop the server or delete its database and, indeed, does not communicate at all with the server. You should apply this method only if the replica has died irrevocably.

If you have forcibly deleted a replica and the replica later resumes operation, you should use the **reset** command in **rgy_admin** to stop the server and delete its database, since the forced deletion has made all other registry replicas unaware of its existence.

Procedure 4–12. *Forcibly Deleting a Registry Replica*

You must be an owner of the registry in order to perform Task 2 of this procedure.

Task 1: Set the default host to the current master site

Invoke **rgy_admin**. Set the default host to the current master registry site by issuing the **set -m** command.

Task 2: Delete the replica

Delete the replica by issuing the following **rgy_admin** command:

`rgy_admin:` **delrep** *//regsite* **-force**

where *//regsite* is the site of the replica to be deleted.

4.11 The import_passwd Command

The **import_passwd** command creates entries in the Apollo registry based on information in UNIX password and group files. It provides a method of ensuring consistency between the Apollo registry and the UNIX password and group entries. Consistency is crucial for most forms of file sharing between Apollo systems and other UNIX systems.

Use this command when you are

- Attaching Apollo node(s) to a foreign network
- Attaching foreign node(s) to an Apollo network
- Connecting Apollo and foreign networks

4.11.1 How import_passwd Processes

When **import_passwd** processes, it compares the foreign group and password file entries to the Apollo registry entries. It can find two types of conflicts:

- **Name Conflicts.** These conflicts arise when the same name string is defined in the Apollo registry and the foreign system. "Joe 102" and "Joe 555" are an example of such a conflict. The duplicate name may represent the same user or two different users.
- **UNIX ID Conflicts.** These conflicts arise when the same UNIX ID is defined in the Apollo registry and the foreign system for users with different names. "Joe 102" and "Ann 102" are an example of such a conflict.

These conflicts can be found separately, as in the examples above, or together. For instance, an Apollo entry of "Joe 102" and a foreign entry of "Joe 102" are in conflict. Unless they represent the same user, one of the entries must be changed.

As **import_passwd** processes, it performs the following steps in sequence:

1. It puts the Apollo registry in maintenance mode and reads the foreign group and password files.
2. It compares the foreign group file entries to the Apollo group entries. If there are no conflicts, it creates Apollo group registry entries corresponding to the foreign groups. (Section 4.11.3 describes what happens if there are conflicts.) Note that the members of the groups are not added at this time, but in Step 4.
3. It compares the entries in the foreign password file to the Apollo person and account entries. Again, if there are no conflicts, it creates Apollo person and account entries corresponding to the foreign file.
4. If there are members in the foreign groups handled in Step 2, it adds them to the appropriate group in the Apollo registry.
5. It then prompts for whether or not to make the changes. If you specify that the changes should be made, it saves the changes to the registry.

4.11.2 Other Entries Created by import_passwd

The **import_passwd** tool modifies only person names, person IDs, group names, group IDs, group members, and account passwords. It does not modify any of the additional information in the registry.

For example, assume you have a foreign password entry for user **jack** and group **staff** and an Apollo account entry of **jack.staff.none**. You run **import_passwd** with the **-i** option. This option tells **import_passwd** to consider the entries identical. The home directory specified in the foreign network is **/usr/u/jack**; the home directory specified in the Apollo network is **//gimli/jack**. The **import_passwd** tool will not change the Apollo registry to match the foreign home directory. The **jack.staff.none** entry in the Apollo registry will have a home directory of **//gimli/jack**, not **/usr/u/jack**.

If **jack.staff.none** did not exist in the Apollo registry, **import_passwd** would create a new registry entry. For the additional information, it assigns the following values:

For Person and Group Entries:

- **fullname** = " (that is, empty).
- **owner** = Same as the owner of the domain as listed in the registry properties (that is, the owner for new person entries is set to Person Owner, and the owner for new group entries is set to Group Owner.)
- **alias/primary** = Primary for first entry; alias for subsequent ones.
- **projlist_ok** (for groups only) = Yes.
- **passwd** = For groups only, taken from the foreign group file.
- **membership list** = For new groups only, all persons listed in the foreign group file and all persons with accounts in the foreign password file with that group.

For Account Entries:

- **abbreviation** = Shortest possible abbreviation that does not conflict with pre-existing Apollo accounts.
- **account_valid** = True.
- **gecos** = Same as foreign password file.
- **homedir** = Same as foreign password file.
- **shell** = Same as foreign password file.
- **passwd** = Same as foreign password file. Note that you must modify or reset imported passwords before user authentication is possible and for the account to be usable in a pre-SR10 registry.
- **passwd_dtm** = Date and time **import_passwd** was run.
- **passwd_valid** = True.

4.11.3 Resolving Conflicts

When you use **import_passwd**, you must decide how to resolve the conflicts it will encounter. The **import_passwd** command provides a number of options to help you. If the conflict cannot be resolved even with the option instructions, **import_passwd** will prompt you for resolution. The options are described in the following subsections.

The Identical User Option

The **-i** option lets you specify that duplicate names are not in conflict but, instead, represent the same identity. When **import_passwd** finds duplicate name entries, it processes them as though they are the same user. If you do not use the **-i** option, **import_passwd** will prompt you to resolve the name conflict.

The Favored Entry Option

The **-a** (favor Apollo) and **-f** (favor foreign) options let you specify whether the Apollo or the foreign entry is the favored entry. A favored entry is retained as is. You are prompted to modify non-favored entries.

For example, suppose you run **import_passwd** with the **-a** (favor Apollo) option and without the **-i** (identical user) option. During processing, the program encounters an Apollo entry of **joe 555** and a foreign entry of **joe 102**. Because the Apollo entry is favored, **joe 555** will be retained in the Apollo registry, and you will be prompted for a new UNIX ID (a new name) for **joe 102**.

The **import_passwd** command also uses the favored entry to resolve UNIX ID conflicts. For example, suppose you are running **import_passwd** with the options described above. During processing, it encounters an Apollo entry of **joe 555** and a foreign entry of **ann 555**. Because the Apollo entry is favored, **import_passwd** prompts you for a new UNIX ID for "ann."

Be aware, however, that Apollo reserved entries cannot be modified. (Table 4-2 list the Apollo reserved entries.) The **import_passwd** command will not modify a reserved entry even if it is the non-favored entry. For example, suppose you are running **import_passwd** with the foreign entry as the favored entry. During processing, it encounters a foreign group entry of **misc 0** and an Apollo group entry of **wheel 0**. Because group **wheel 0** is a reserved Apollo entry, you will be be prompted to modify the foreign entry, even though it is the favored entry.

Conflict Summary

Table 4–3 summarizes the effects of the identical user and favored entry options.

Table 4–3. Effects of Identical User and Favored Entry Options

Options Used	Foreign Entry	Apollo Entry	Result in Apollo Registry	Comments
-i, -a	joe 102	joe 555	joe 555	Name collision. Retain Apollo entry.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
-i, -f	joe 102	joe 555	joe 102	Name collision. If 102 already exists in Apollo, prompt for new UNIX ID for that Apollo entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.
-a	joe 102	joe 555	joe 555	Name conflict. Request new name for joe 102, and if 102 is already defined in Apollo, a new UNIX ID as well.
	joe 102	ann 102	ann 102	UNIX ID conflict. Request new UNIX ID for joe.
-f	joe 102	joe 555	joe 102	Name conflict. Request new name for joe 555, and if 102 is already defined in Apollo, prompt for a new UNIX ID for that Apollo entry.
	joe 102	ann 102	joe 102	UNIX ID conflict. Request new UNIX ID for ann.

4.11.4 The import_passwd Syntax

This section presents the **import_passwd** command and describes the command arguments and options.

import_passwd [-i] [-a | -f] [-c] [-o *org***] -s** *pathname* **[-v]**

-i	Identical name strings are not in conflict, but represent the same identity.
-a (D)	Favor Apollo entries.
-f	Favor foreign entries.
-c	Run in check mode: process the command showing conflicts, but make no changes to the files.
-o *org*	*org* is the name of the Apollo organization to be used for all imported account entries. The default is the organization named "none."
-s *pathname*	*pathname* is the path to the directory containing the foreign password and group files to be imported.
-v	Run in verbose mode: generate a verbose transcript of all activity.

4.11.5 Using import_passwd

This section describes conflicts you may encounter when running **import_password.**

Using Check Mode

You should first run **import_passwd** in check mode using the **-c** option. In this mode, **import_passwd** simulates the results of a processing run showing the conflicts that will be be encountered when **import_passwd** is run without the **-c** option.

Check mode gives you a good idea of the quantity and complexity of the potential conflicts. However, check mode does not make any changes to the file. When you run without the **-c** option and make changes to resolve conflicts, these changes can in turn create further conflicts not readily apparent in check mode.

If you encounter numerous conflicts in check mode, you may find it more efficient to manually edit either the registry or the UNIX group and password files to resolve some obvious conflicts before you run **import_passwd.**

Answering Prompts

When you run **import_passwd**, you may be prompted for names and numbers (UNIX IDs). The following conventions apply to your answers to these prompts:

- Names must
  - Begin with a lowercase letter
  - Contain *only* lowercase letters, digits, or underscore characters
  - Not exceed 32 characters in length, unless your system imposes UNIX restrictions, in which case, they cannot exceed 8 characters in length

  If your system imposes UNIX name restrictions, you should carefully evaluate the effect of these restrictions on existing entries in the files against which you are running **import_passwd**. Some of the entries may be longer that 8 characters.
- Numbers must range between 0 and 65535, inclusive. (We suggest that you not use numbers under 100, since these may be reserved in future releases.)

If you enter a name or number in an incorrect format, **import_passwd** will ignore your entry and re-prompt.

Processing Prerequisites

The Apollo registry must exist before you can use **import_passwd**. If you are simply adding a few Apollo nodes to a foreign network, you can create a new, but empty, registry to meet this requirement. Once the registry exists, the registry server must be running and you must be logged on as root.

Synchronizing Apollo UNIX IDs

If **import_passwd** makes any changes to Apollo UNIX IDs, you must run the **/etc/syncids** tool on every Apollo node in your network. This tool ensures that the changes are synchronized with the UNIX IDs stored on disk. On nodes that have more than one disk volume, you must run **syncids** on each volume.

If your Apollo registry is replicated before you run **syncids**, you should use the **lrep -state** command in the **/etc/rgy_admin** tool to verify that all slave registry servers are up to date. This command shows the time stamps of the most recent changes to each replica of the registry database. You should not run **syncids** until the time stamps for all slave servers match the time stamp for the master.

See the reference documentation for **syncids** and **rgy_admin** online or in the *Command Reference* manual for your environment.

Synchronizing Foreign UNIX IDs

If any of the conflicts found by **import_passwd** involve reserved information in the Apollo registry, you may have to resolve conflicts by changing names and IDs on your foreign systems. You also may have to change names and IDs on your foreign systems if you run **import_passwd** with the **-a** (favor Apollo) option. In that event, you must then use the **chown** and **chgrp** utility to update the UNIX IDs on all affected files and directories in the foreign file systems.

4.12 A Sample import_passwd Session

This subsection illustrates a simplified **import_passwd** session. The session uses the foreign group file and password files shown in Figure 4-3 and the Apollo group file and password entries shown in Figure 4-4.

Foreign Group File Entries

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
daemon::12:root,daemon
tgroup::35:
```

Foreign Password File Entries

```
root::0:1:0000-Admin(0000):/:
daemon::1:1:0000-Admin(0000):/:
bin::2:2:0000-Admin(0000):/bin:
sys::3:3:0000-Admin(0000):/usr/src:
adm::4:4:0000-Admin(0000):/usr/adm:
uucp::5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp::10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/
uucico
rje::18:18:0000-rje(0000):/usr/rje:
trouble::70:1:trouble(0000):/usr/lib/trouble:
lp::71:2:0000-lp(0000):/usr/spool/lp:
setup::0:0:general system administration:/usr/admin:/bin/rsh
powerdown::0:0:general system administration:/usr/admin:/bin/rsh
sysadm::0:0:general system administration:/usr/admin:/bin/rsh
checkfsys::0:0:check diskette file system:/usr/admin:/bin/rsh
makefsys::0:0:make diskette file system:/usr/admin:/bin/rsh
mountfsys::0:0:mount diskette file system:/usr/admin:/bin/rsh
umountfsys::0:0:unmount diskette file system:/usr/admin:/bin/rsh
```

Figure 4-3. Foreign Group and Password Entries

Group Entries

```
wheel::0:
daemon::1:
none::2:
backup::3:user
locksmith::4:
login::5:
mail::6:bin
bin::7:root
server::8:
sys::9:root
staff::10:
sys_admin::11:user
sys_proj::12:
tgroup::35:
```

Password Entries

```
root:sq1RclUrrb1L6:0:10::/:
daemon:sq1RclUrrb1L6:1:2::/:
none:sq1RclUrrb1L6:2:2::/:
user:sq1RclUrrb1L6:3:2::/:
lp:sq1RclUrrb1L6:4:7::/:
sys_person:sq1RclUrrb1L6:5:2::/:
admin:sq1RclUrrb1L6:6:2::/:
uucp:sq1RclUrrb1L6:7:2::/usr/spool/uucppublic:
bin:sq1RclUrrb1L6:8:7::/:
```

Figure 4–4. Apollo Group and Password Entries

The following subsections describe a session in which you perform the following tasks: invokethe command, examine the group entries, examine the password file, add members to groups, and update the registry.

Phase 1: Invoking import_passwd

1. In the sample session, the following form of the **import_passwd** command is entered at the shell prompt:

   ```
   % import_passwd -s sys5.3_tapes/adm -i -v
   ```

   This command specifies the following:

   - Identical names represent the same identify (**-i**)
   - The pathname of the foreign file (**-s sys5.3_tapes/adm**)
   - Apollo entries should be favored (default of **-a**)
   - Not running in check mode (**-c** not specified)
   - Do run in verbose mode (**-v**)

2. The system puts the registry in maintenance mode, reads the foreign group and password files, and begins to create Apollo group entries in the Apollo registry:

```
Preparing registry...
Preparing foreign files...
Creating Group entries from foreign group file...
```

Phase 2: Examining the Group Entries

If you examine Tables 4–2 and 4–3, you can see that the name **root** in the foreign and the name **wheel** in Apollo both have UNIX IDs of 0.

3. The **import_passwd** command tells you this and, since Apollo entries are favored, prompts for a new UNIX ID for the foreign entry:

```
root::0:root
?(import_passwd) Group - UNIX id conflict
Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
Please enter new UNIX id for Foreign Group 'root' 0: 100 <RETURN>
>> Adding pgo entry for: root 100
```

The value **100** is entered as the new UNIX ID for the foreign group entry. Note that **import_passwd** displays the foreign entry as it examines it (in the sample above, **root: :0:root**). And, because it is running in verbose mode, **import_passwd** describes the actions it is taking. Each such description is prefaced with the symbols >>.

If you were running **import_passwd** in check mode, you would not be prompted for a UNIX ID. Instead, you would be informed of the need and processing would continue. The message would look like

```
root::0:root
?(import_passwd) Group - UNIX id conflict
Foreign Group: "root" 0  Apollo: "wheel" 0 (reserved).
    need new UNIX id for Foreign Group
      (UNIX id for Apollo "wheel" is currently 0)
```

4. The **import_passwd** command then finds another UNIX ID conflict:

```
other::1:
?(import_passwd) Group - UNIX id conflict
Foreign Group: "other" 1  Apollo: "daemon" 1 (reserved).
Please enter new UNIX id for Foreign Group 'other'1: 101<RETURN>
>> Adding pgo entry for: other 101
```

Note that **import_passwd** tells you that the UNIX ID of **1** is reserved in Apollo. Because **1** is reserved, even if you had run **import_passwd** with the foreign entry favored, you would still be prompted for a new entry for the foreign UNIX ID. The **import_passwd** command will not change reserved entries.

5. If **import _passwd** finds no conflicts, it displays the group entries as it processes them and, because it is running in verbose mode, the actions it is taking:

```
bin::2:root,bin,daemon
>> Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id

   sys::3:root,bin,sys,adm
   >> Foreign Group: "sys" 3  Apollo: "sys" 9 (reserved)
   >>   -i specified - retaining Apollo UNIX id
```

As it continues through the foreign group file, **import_passwd** finds two other UNIX ID conflicts: foreign entries **adm 4** and **rje 8** and Apollo entries **locksmith 4** and **server 8**, respectively.

Phase 3: Examining the Password File

6. **Import_passwd** then proceeds to create Apollo person and account entries from the foreign password file. It displays:

```
Creating Person entries and Accounts from foreign passwd file...
```

7. The first two entries are processed with no conflicts:

```
root::0:1:0000-Admin(0000):/
>> Foreign Person: "root" 0  Apollo: "root" 0 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for root.other.none.

daemon::1:1:0000-Admin(0000):/
>> Foreign Person: "daemon" 1  Apollo: "daemon" 1 (reserved)
>>   -i specified and UNIX ids match - no change necessary
>> Adding account for daemon.other.none.
```

The **import_passwd** command displays the names of the registry accounts it is creating. If you had changed a person or group name to resolve a previous name conflict, the new name would be displayed. For example, suppose that you had a foreign and an Apollo entry of **joe.other.none**. During **import_passwd** processing, you changed the name of the Apollo **joe** to **smith**. When **import_passwd** created the Apollo person and account entries, a verbose display would look like:

```
joe::1:1:0000-Admin(0000):/
>> Adding account for smith.other.none.
```

The foreign entry is displayed first with its correct name, **joe**. The account being added to the Apollo registry is displayed by its new name, **smith**. **Import_passwd** keeps track of changes that are made and the relationships that existed before the changes were made.

8. The third entry produces the following warning message:

```
bin::2:2:0000-Admin(0000):/bin
   >> Foreign Person: "bin" 2  Apollo: "bin" 8 (reserved)
   >>   -i specified - retaining Apollo UNIX id
   >> Adding account for bin.bin.none .
   (WARNING) "bin.bin.none": Account already exists and is re-
   tained unchanged.
```

This message is notifying you that the **bin.bin.none** account exists in the Apollo registry. The Apollo account will retain its information (that is, abbreviation, account_valid, gecos, passwd, shell, home directory, passwd_dtm, and passwd_valid), even though the foreign account may have information that differs from the Apollo account.

9. Processing continues. The **import_passwd** command discovers two more UNIX ID conflicts (foreign **sys 3** and Apollo **user 3**; foreign **adm 4** and Apollo **lp 4**). Then **import_passwd** finds an un-named group in the foreign password file and a UNIX ID conflict.

   First **import_passwd** prompts you for a group name:

```
uucp::5:5:0000-uucp(0000):/usr/lib/uucp
>> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id
?(import_passwd) Foreign Group 5 is unnamed.
    (UNIX id also conflicts with Apollo "login" (reserved) ).
Please enter a name for Foreign Group 5: group_105 <ENTER>
```

   In the sample session, **group_105** is entered as the new name.

   Note that if you were running in check mode, **import_passwd** would supply a name for the group in order to keep processing. The prompt would look like:

```
uucp::5:5:0000-uucp(0000):/usr/lib/uucp
>> Foreign Person: "uucp" 5  Apollo: "uucp" 7 (reserved)
>>   -i specified - retaining Apollo UNIX id
?(import_passwd) Foreign Group 5 is unnamed.
    (UNIX id also conflicts with Apollo "login" (reserved) ).
    Temporarily using the name of "group_5"
```

   Then, **import_passwd** asks you for the new UNIX ID for the group:

```
?(import_passwd) Group - UNIX id conflict
Foreign Group: "g105" 5  Apollo: "login" 5 (reserved).
Please enter new UNIX id for Foreign Group 'g105' 5: 105 <ENTER>
>> Adding pgo entry for: g105 105
>> Adding account for uucp.g105.none.
```

Phase 4: Adding Members to Groups

10. When **import_passwd** completes processing of the foreign password file, it adds the members to the groups:

```
Adding memberships from foreign group file...
root::0:root
  Member: root
other::1:
bin::2:root,bin,daemon
  Member: root
  Member: bin
  Member: daemon
```

    As it adds the members, **import_passwd** displays them. If there are no members added (as in group **other::1:**), none are displayed. If you are not running in ver-

bose mode, only the names of the groups are displayed as they are processed, not the members being added.

11. When **import_passwd** processes the **rje::8:rje,shger** entry, it displays the following message:

```
rje::8:rje,shqer
  Member: rje
  Member: shqer
?(import_passwd) Cannot add member - Person is unknown or invalid
```

In foreign UNIX systems, you can add members to groups even if the person has not been added as a person. In Apollo systems you cannot. The **import_passwd** command has found a member of a foreign group who has not been added as a person. Therefore, **import_passwd** will not add that member to the group.

Phase 5: Updating the Registry

12. Processing continues until **import_passwd** completes adding members to groups, it recaps the results of its processing:

```
0 name conflicts, 8 UNIX id conflicts, 3 unnamed groups,1 error,
2 warnings.
```

And then asks if you want to save the changes:

```
Import operation successful.  Update registry [y/n]?  y <RETURN>
```

In this sample session, the answer is Y (yes).

13. The **import_passwd** command then saves the changes and completes processing:

```
Closing registry...
Closing foreign files...
Done.
```

Because the **-i** option was specified in the command used in the sample session, no name conflicts were encountered. A sample of the prompt resulting from a name conflict is shown below:

```
bin::2:root,bin,daemon
?(import_passwd) Group - Name conflict
Foreign Group: "bin" 2  Apollo: "bin" 7 (reserved)
Please enter new name for Foreign Group "bin": fbin <ENTER>
>> Adding pgo entry for: fbin 2
```

In the sample above, a new name of **fbin** is assigned in the Apollo registry to entry representing the foreign group **bin**.

4.13 Local Registries

Each node can have a local registry that contains information about the node's most recent users and the date and time when they last logged in. If you log in to a node and no registry server is available on the network, the operating system checks the node's local registry for information about your account. If the information exists in the local registry, the system allows you to log in with that account; if not, it refuses to log you in. If both the network registry and the local registry are unavailable, the system will always allow you to log in with the account name **user.none.none**.

4.13.1 Setting Up the Local Registry

The operating system creates a local registry the first time a user logs in to a node, provided a registry server is running somewhere on the network. Thereafter, the local registry is updated every time someone logs in to the node. Users can edit some of the information in a node's local registry by invoking the **edrgy -l** command on that node.

4.13.2 Running a Small Network without Registry Servers

If you have a small network consisting of one or two nodes with no more than 10 accounts, it is possible to operate your network by using only local registries. You need to run **rgyd** while you create accounts, then copy account information into the local registry on each node, as follows:

1. Start a **rgyd** process on one node and use **edrgy** to create the accounts you need.
2. Use the **copy** command in **edrgy** to add account information to the local registry. The command **copy %.%.%** will fill the local registry to capacity.
3. Use the **stop** command in **rgy_admin** to terminate the **rgyd** process.

The accounts you created now have access to the nodes via the local registry. Of course, if you wish to change any account information, you must restart **rgyd** to make the changes.

(Even without local registries, all reserved entries are available to the operating system, so users on your network can log in as **user.none.none**.)

Chapter 5

Protection of Files and Directories

Contents

Chapter 5

Protection of Files and Directories

This chapter describes how SR10 Domain/OS controls access to files and directories. Domain/OS implements the UNIX protection model and extends it with an Access Control List (ACL) model. Under Domain/OS, UNIX protection operates as you would expect on other UNIX systems, while the ACL mechanism provides greater flexibility in controlling access to objects.

Section 5.1 describes UNIX protection. Section 5.2 describes the ACL model and the ways in which it extends the UNIX model. Section 5.3 explains how Domain/OS derives UNIX modes from ACLs. (If you use only vanilla UNIX protection, without any of the ACL extensions, you can skip Sections 5.2 and 5.3.) The remaining sections discuss protection in the context of system administration.

NOTE: The SR10 protection-checking mechanism can operate over the network with the mechanism released at SR9.7, but not with those released prior to SR9.7. Therefore, if your site is upgrading to SR10 over a period of time and you want to share files between SR10 and non-SR10 nodes, you should ensure that all non-SR10 nodes have system software of SR9.7 or later vintage. For information about transition to SR10 and about operating an environment with SR9.7 and SR10 nodes, see *Making the Transition to SR10 Operating System Releases*. For information about installing software, see *Installing Software with Apollo's Release and Installation Tools*.

5.1 UNIX Protection

The UNIX protection scheme is based on owner and group IDs. Every file system object has associated with it an owner ID and a group ID. Every process has four IDs: a real owner ID, an effective owner ID, a real group ID, and an effective group ID. Real IDs always represent the actual owner and group of the process; effective IDs are used by the system for checking rights.

5.1.1 Protection Modes

Each file and directory has permissions for three categories of users: owner, group, and others. These permissions are represented as a UNIX **mode**, an octal number constructed from the logical OR of the following bits:

4000	set user ID on execution
2000	set group ID on execution
1000	sticky bit
0400	read by owner
0200	write by owner
0100	execute by owner
0040	read by group
0020	write by group
0010	execute by group
0004	read by others
0002	write by others
0001	execute by others

For example, mode 640 allows the owner to read and write, the group to read only, and others no access at all.

The UNIX command **ls**, when given the **-l** option, reports an object's octal mode in the form

```
-rwxrwxrwx
```

where each rwx sequence stands for the set of permissions available to the owner, the group, and others, respectively. (The initial hyphen indicates that the object is an ordinary file. For a directory, the initial character is the letter "d"; for a symbolic link, it is the letter "l".) Mode 640, for example, appears as

```
-rw-r-----
```

Table 5-1 shows these rights and their meanings for files and directories.

Table 5-1. UNIX Permissions

	File	Directory
r	Read	List entries
w	Write	Add, delete, or change entries
x	Execute	Search

5.1.2 The setuid and setgid Bits

The 4000, 2000, and 1000 bits in a protection mode have meaning only for executable files. They affect the behavior of a file when a user executes it.

The 4000 bit is the **setuid** bit. If you execute a file that has this bit on, the operating system sets the effective user ID of the process to the owner ID of the file. Similarly, the 2000 bit is the **setgid** bit. It sets the effective group ID to the group ID of the file.

The **ls** command reports the setuid (or setgid) bit with the letter "s" in place of the "x" for execution by owner (or group). Though **ls** displays them in place of the "x," these bits do not affect permission to execute the file. The operating system always checks the "x" bit.

For example, if a program has the protection mode 6711, the owner ID **pooh**, and the group ID **bears**, **ls -l** reports its protection as

```
-rws--s--x
```

The program always executes as though it were invoked by user **pooh** and group bears.

The 1000 bit is the **sticky** bit. On Domain systems, this bit has no effect.

5.1.3 Checking Permissions

When a process attempts to access a file system object, the operating system checks permissions in order: first owner, then group, and finally others. The first matching permissions apply. Permissions for the owner apply to any process whose effective user ID matches the object's owner ID. Group permissions apply to any process whose effective group ID matches the object's group ID and whose user ID does not match the object's owner ID. Permissions for others apply to all other processes.

For example, suppose the file **tarts** has the owner ID **jack**, the group ID **hearts**, and the protection mode 640. With the **-l** and **-g** options, **ls** will show complete protection information for **tarts**:

```
% ls -lg tarts
-rw-r-----   1 jack     hearts       3474 Feb 29 10:54 tarts
```

Processes with an effective user ID of **jack** will have both read and write permissions for **tarts**. Other processes will have read permission if their effective group ID is **hearts** and otherwise will have no access at all.

The **/etc/passwd** file specifies a default group for each user. Additional groups can be specified in the **/etc/group** file. Under SysV, users can invoke the **newgrp** command to change their effective group IDs. Under BSD, a user effectively has membership in several groups at once: the default group specified in the **passwd** file and any others specified in the **group** file.

In Domain/OS, the PROJLIST environment variable determines which scheme for group membership applies. PROJLIST is set by default in the BSD operating environment. In other operating environments you can set it optionally. (See Chapter 4, Subsection 4.2.4 for details.)

Because the operating system always applies the first matching permissions, it is possible for owner of a file to have fewer permissions than other users on the system. Mode 477, for instance, gives the owner read rights only, but gives everyone else all rights.

5.1.4 Assigning and Changing Permissions

A newly created object inherits its owner and group IDs from the process that creates it. (Except that under 4.3BSD, an object inherits the group ID from the parent directory, the directory where the object is created.)

The permissions for a newly created object can be specified by the creating process, through the *mode* argument to system calls such as **mkdir** or **open**. System calls that do not take a *mode* argument create objects with mode 777. In both cases, the permissions specified by the creating call are then filtered through the **umask** of the process. The umask is a bitmask that specifies permissions to be disallowed for any objects created by the process. For example, if a process with a umask of 022 creates an object via an **open** call with a *mode* of 770, the object will have a permissions mode of 750.

Only an object's owner or the super–user can change the IDs and permissions associated with the object. (The super–user, which we discuss in Subsection 5.4.1, has rights outside the normal protection model.) On vanilla 4.3BSD systems, only the super–user can change the owner ID of an object. However, Domain/OS BSD does not implement this restriction.

5.1.5 Utilities

This subsection surveys the UNIX utilities for inspecting and modifying protection. See the *BSD Command Reference* and the *SysV Command Reference* for detailed descriptions.

The **umask** command (built in to the Aegis shell as well as the UNIX shells) sets or displays the value of the umask. The **chmod** command changes the permissions mode of a file or directory; **chown** changes the owner ID, and **chgrp** command changes the group ID. To display the protection of an object, you can use **ls** with the **–l** and **–g** options.

Figure 5–1 illustrates the use of these utilities. In this example, files are created via the **touch** command, which uses the **creat** system call with a *mode* of 666.

```
% umask
0
% touch magritte
% ls -lg magritte
-rw-rw-rw-   1 mk        none            0 Feb  5 14:48 magritte
% umask 022
% touch dali
% ls -lg dali
-rw-r--r--   1 mk        none            0 Feb  5 14:49 dali
% chmod 664 dali
% chgrp dds dali
% ls -lg dali
-rw-rw-r--   1 mk        dds             0 Feb  5 14:49 dali
```

Figure 5-1. UNIX Protection Utilities

The *BSD Programmer's Reference* and the *SysV Programmer's Reference* describe system calls pertaining to protection, including **stat**, **chmod**, **chown**, and **umask**.

5.2 Domain/OS Protection

The SR10 Domain/OS protection model incorporates UNIX semantics and extends them with Access Control Lists (ACLs). The Domain/OS model is thus a superset of the UNIX model, as shown in Figure 5-2.

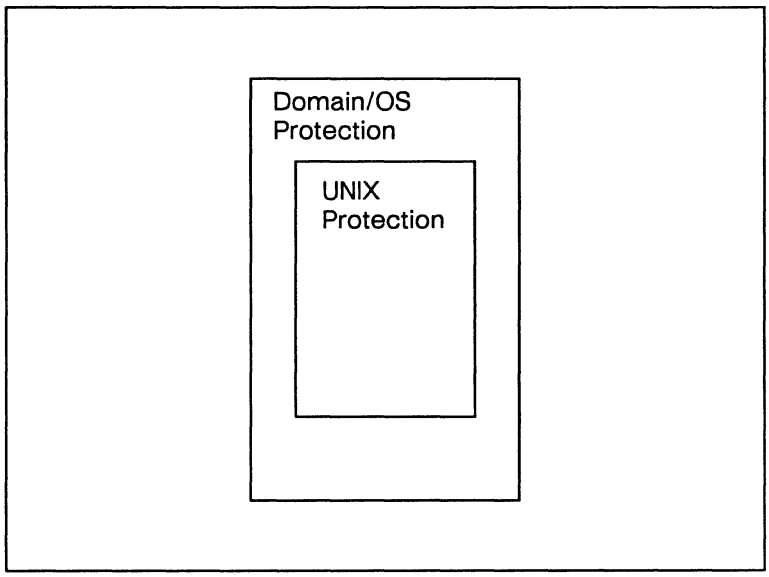


Figure 5-2. Domain/OS and UNIX Protection

5.2.1 ACL Structure

Each file system object has an ACL that defines who can perform what operations on that file or directory. An ACL contains a number of entries. Each ACL entry has two parts: a **Subject Identifier (SID)** and a set of **access rights.**

Subject Identifiers

An SID is a specifier consisting of fields for a person, a group, and an organization. In ACL entries, each field of the SID can be either a name or a wildcard (%). Section 4.2 describes SIDs in more detail.

Access Rights

Access rights describe the extent of an SID's access to the associated object. Table 5-2 lists the access rights for files and directories.

Table 5-2. Domain/OS Access Rights

Access	File	Directory
r	Read	List entries
w	Write	Add, delete, or change entries (including links)
x	Execute	Search
p	Change protection	Change protection
k	Keep (prevents deletion or change of name)	Keep (prevents deletion or change of name)

The **r**, **w**, and **x** rights are like their UNIX counterparts.

In vanilla UNIX systems, only the owner of an object or the super-user can change the object's protection. However, in Domain/OS, you can use the **p** access right to grant any SID permission to change the object's protection.

The **k** right in an ACL entry denies the specified SID the right to delete the object or to change its name, even if that SID has write permission for the parent directory. Under traditional UNIX semantics, all objects in a writable directory are deletable. However, in some situations, you want a directory to be writable, but you want to protect a particular file or subdirectory from deletion. You can establish this protection by setting the **k** right in the ACL for the file or subdirectory.

If the ACL entry for an object contains both **p** and **k** rights, the specified SID can delete the object by using the **-f** option to either of the Aegis commands **dlf** and **dlt**. However, the UNIX command **rm** will not delete the object unless invoked by the super-user.

5.2.2 Types of ACL Entries

An ACL must contain four **required entries**; in addition, it can contain **extended entries**. The required entries, which express UNIX protection information, are stored in the i-node of each file or directory to provide rapid access for operations like **stat** and **chmod**.

Required Entries

Every ACL contains at least four entries: one each for person, group, organization, and world. These are the four required entries in an ACL. The person, group, and world entries correspond to the UNIX concepts of owner, group, and other. The organization entry does not have a UNIX analog.

A required ACL entry can be designated as an **ignored entry**. When the operating system checks permission to access the object, it ignores the entry. The **acl** command displays the entry with the word "ignored" in place of the access rights.

The SIDs in the four required entries must be of the forms *person.%.%*, *%.group.%*, *%.%.organization*, and *%.%.%*.

Extended ACL Entries

In addition to the four that are required, an ACL can contain up to 22 extended entries. As we will explain in Subsection 5.3.2, the operating system applies a mask to the rights specified in extended entries when it checks permissions.

The SIDs in extended entries can specify any combination of *person*, *group*, and *organization*, provided they do not duplicate any SIDs in the required entries. An extended entry cannot be ignored.

An Example

Figure 5-3 shows the ACL for a file as reported by the **acl** command. The left column shows SIDs; the right column shows the access rights associated with each SID.

NOTE: All examples in this chapter use the **acl** utility. If you are working in a purely BSD environment (that is, you do not have a **/com** directory), use the **lsacl** command (see the **lsacl** manual page).

```
% acl bar
Acl for bar:
Required entries
 mk.%.%                          prw--
 %.dds.%                         -rw--
 %.%.r_d                         [ignored]
 %.%.%                           -r---
 Extended entry rights mask:     -r---
Extended entries
  %.backup.%                      -r---
  curly.stooges.%                 -----
```

Figure 5-3. An ACL

5.2.3 Types of ACLs

There are four types of ACL: the file ACL, the directory ACL, the initial default file ACL, and the initial default directory ACL. The first type is associated with files; the other three are associated with directories. ACLs of all four types must contain the required entries for owner, group, organization, and world.

The **file ACL** controls access to the file with which it is associated.

The **directory ACL** controls access to the directory with which it is associated.

The **initial file ACL** determines the protection for files created within a directory.

The **initial directory ACL** determines the protection for subdirectories created within a directory.

Initial ACLs determine the protection assigned to newly created files and directories. Subsection 5.2.5 discusses in detail the role of initial ACLs.

5.2.4 How ACLs are Interpreted

The operating system associates each process with an SID. When a process attempts to access a file or directory, the operating system compares the process' SID with the entries in the ACL for the file or directory; it applies the access rights specified in the first entry whose SID matches the process' SID. If the rights allow the requested mode of access, the process gains access; if not, access is denied.

Because the first matching ACL entry determines a process' access to an object, the order of ACL entries is important. In general, ACL entries are ordered from the most specific to the least specific, with special precedence for required entries. The person field is more specific than the group field, which in turn is more specific than the organization field.

Figure 5–4 illustrates the rules for ordering of ACL entries.

```
mk.%.%              prwx-          (required)
mk.dds.r_d          prwx-
mk.dds.%            prwx-
mk.%.r_d            prwx-
user.unix.%         -r---
user.%.%            -r---
%.unix.%            -rwx-          (required)
%.dds.r_d           -rwx-
%.dds.%             -rwx-
%.%.r_d             -r-x-          (required)
%.%.legal           -r---
%.%.%               -----          (required)
```

Figure 5–4. Effective Order of Entries in an ACL

Note that Figure 5–4 shows the effective order of ACL entries but does not correspond to the output of the **acl** command; **acl** displays extended entries separately from required entries. Figure 5–5 shows the same ACL, as **acl** would display it.

```
% acl foo
Acl for foo:
Required entries
 mk.%.%                          prwx-
 %.unix.%                        -rwx-
 %.%.r_d                         -r-x-
 %.%.%                           -----
 Extended entry rights mask:     prwx-
Extended entries
  mk.dds.r_d                      prwx-
  mk.dds.%                        prwx-
  mk.%.r_d                        prwx-
  user.unix.%                     -r---
  user.%.%                        -r---
  %.dds.r_d                       -rwx-
  %.dds.%                         -rwx-
  %.%.legal                       -r---
```

Figure 5–5. An ACL as Displayed by the acl Command

5.2.5 How ACLs are Assigned to New Files and Directories

When a process creates a file or directory, the operating system assigns an ACL to the new object. Three factors govern the assignment of ACLs:

- The protection mode, if any, specified by the program creating the object
- The **umask**, if any, of the process creating the object
- The initial ACLs of the **parent directory**, the directory in which the object is created

Creation Mode and the umask

Some UNIX system calls, including **open** and **mkdir**, take a *mode* argument that allows you to specify a protection mode for a newly created object. Some other UNIX system calls and all Aegis system calls specify the widest possible permissions, allowing all forms of access for all users. In either case, this **creation mode** is filtered through the umask of the process, as Subsection 5.1.4. explains.

In Domain/OS, the low-order octal digit in the *mode* and the umask applies both to the required organization entry and to the **%.%.%** entry. The middle digit applies to the required group entry. The high-order digit applies to the required person entry.

The permissions that result from application of the umask to the creation mode are either assigned to the new object or overridden by entries in the initial file or directory ACL of the parent directory. We refer to the first mechanism as "inheritance from the process" and the second as "inheritance from the parent directory." The choice between these mechanisms is specified within the initial ACL itself.

Initial ACLs

Domain/OS allows the SID and/or the rights of a required ACL entry to be inherited either from the creating process or from the parent directory. Inheritance is specified as part of the initial ACLs of the parent directory and can apply to the SID, to the rights, or to both parts of an entry.

Figure 5-6 shows an initial file ACL that implements BSD protection semantics. The rights for the required organization entry are ignored. All other SIDs and rights are inherited from the creating process except for the group SID, which derives from the parent directory.

```
% acl -if figaro
Initial (default) acl for files created under figaro
Required entries                                          For the current process:
 [from process]                  [specified by process]   mk.%.%
 %.dds.%                         [specified by process]
 %.%.r_d                         [ignored]
 %.%.%                           [specified by process]
 Extended entry rights mask:     -----
```

Figure 5-6. Initial File ACL Implementing BSD Inheritance

As Figure 5-6 shows, **acl** reports under the heading "For the current process" the particular SIDs that will be inherited if the current process creates an object.

The **edacl** command has options that set inheritance of ownership or permissions in the initial ACLs of a directory. However, you should not ordinarily need to use these options; when Domain/OS software is installed on your node, the initial ACLs of your directories are set to implement the inheritance mechanism appropriate for your environment. The **chacl** command allows you to change easily the inheritance semantics for any particular directory.

5.2.6 Utilities

This subsection surveys the utilities that deal with ACLs.

The Aegis operating environment provides the utilities **acl** and **edacl**. You can use **acl** to display an ACL or to copy an ACL from one object to another. You can use **edacl** to edit ACLs.

In all three of the operating environments, the utilities **chacl, lsacl,** and **cpacl** are available in **/usr/apollo/bin**. These tools offer functionality similar to that of the Aegis **acl** and **edacl** utilities but user interfaces similar to those of the UNIX **chmod, ls,** and **cp** utilities.

If you use BSD or SysV UNIX protection without any of the Domain/OS extensions, you will not need to deal with ACLs at all. See Subsection 5.1.5 for a survey of the UNIX protection utilities.

5.3 How UNIX Modes are Derived from ACLs

This section describes how Domain/OS protection is translated into UNIX protection.

Domain/OS derives the modes that it supplies to UNIX programs and calls from the access rights in the ACL. If the protection of an object involves Domain/OS extensions to the UNIX protection model, the ACL information might not map directly to UNIX modes. In such cases, the operating system will over-represent rather than under-represent the access available when it reports permissions, and it will over-restrict rather than under-restrict access when it sets permissions.

> **NOTE:** The information in this section is important only for those who use the Domain/OS extensions to UNIX protection. Vanilla UNIX protection behaves as you would expect on other UNIX systems.

5.3.1 Permissions for Owner and Group

As Figure 5-7 shows, permissions for the owner and group of an object are those specified in the required ACL entries for the person and group.

```
% acl foo
Acl for foo:
Required entries
 mk.%.%                              prwx-
 %.dds.%                             prw--
 %.%.r_d                             -rw--
 %.%.%                               -----
 Extended entry rights mask:         -rwx-
Extended entries
  user.%.%                            ---x-
  %.osdev.%                           -rw--
  %.backup.%                          -r---
% /bin/ls -lg foo
-rwxrw-rwx  1 mk        dds            0 Nov 10 17:35 foo
```

Figure 5–7. ACL Entries and UNIX Permissions for a File

5.3.2 Permissions for Others

Since ACLs do not provide a direct equivalent to UNIX "other" rights, the operating system derives these rights from information in the ACL. The ACL entries that determine UNIX "other" rights are the required organization entry, the required world entry, and any extended entries—that is, all except the required person and group entries.

To expedite the mapping of ACL information to UNIX rights, the operating system maintains for every file or directory a set of rights that summarizes the permissions allowed by extended entries. These rights are reported in the output of **acl** as the "extended entry rights mask" and in the output of **lsacl -l** as the "extended entry mask." Unless otherwise set by a **chmod** call or command, the mask is the logical OR of all rights in any extended entries.

The remainder of this subsection discusses how the rights mask, the org rights, and the world rights are used when the operating system reports, checks, or sets UNIX rights.

Reporting Permissions

When UNIX "other" rights are required by a command such as **ls** or a call such as **stat**(2), the operating system computes and reports the logical OR of the following:

- The rights in the required *org* entry
- The rights in the required *world* entry
- The extended entry rights mask

In Figure 5–7, for example, **ls** reports "other" rights as **rwx** because each of these rights is available to some SID other than the owner and the group, even though none of those SIDs has all of the rights.

Setting Permissions

The rights mask reflects changes you make via **edacl** or **chacl** to the extended entries in an object's ACL. You can set the mask itself, independently of any ACL entries, via **chmod**.

If you use **edacl** or **chacl** to edit extended entries in the object's ACL, the operating system recomputes the rights mask by taking the OR of all rights in the extended entries. The mask does not change if you edit only required entries.

If you use the **chmod** call or command to change the rights for "others" on the object, the operating system implements the change as follows:

- It sets the rights mask to match the rights you specified for "others."
- It sets "world" (**%.%.%**) rights to those you specified for "others."
- It sets the required *org* entry to be ignored.
- It does not change any extended entries.

As a result, no SID can have any rights not allowed in the **chmod** call or command.

Figure 5–8 illustrates the use of **chmod** to change UNIX group and "other" rights on a file.

Checking Permissions

When it checks permissions for a extended entry, the system takes the logical AND of the rights mask and the permissions in the entry. Thus, SIDs in extended entries are allowed at most the access permitted by the rights mask.

After the **chmod** command in Figure 5–8, for example, **%.osdev.%** and **%.backup.%** can only read **foo**, and **user.%.%** can only execute it. The w in the entry for %.osdev.% is masked.

```
% acl foo
Acl for foo:
Required entries
 mk.%.%                              prwx-
 %.dds.%                             prw--
 %.%.r_d                             -rw--
 %.%.%                               -----
 Extended entry rights mask:         -rwx-
Extended entries
  user.%.%                            ---x-
  %.osdev.%                           -rw--
  %.backup.%                          -r---
% /bin/ls -lg foo
-rwxrw-rwx  1 mk        dds             0 Nov 10 17:35 foo
% /bin/chmod 775 foo
% acl foo
Acl for foo:
Required entries
 mk.%.%                              prwx-
 %.dds.%                             -rwx-
 %.%.r_d                             [ignored]
 %.%.%                               -r-x-
 Extended entry rights mask:         -r-x-
Extended entries
  user.%.%                            ---x-
  %.osdev.%                           -rw--
  %.backup.%                          -r---
% /bin/ls -lg foo
-rwxrwxr-x  1 mk        dds             0 Nov 10 17:35 foo
```

Figure 5–8. Use of chmod to Set Permissions

5.4 Special Groups and Accounts

This section discusses special groups and accounts that you can use in managing Domain systems. These groups and accounts are **reserved**: they are added automatically and permanently to the registry database when the database is initialized. Subsection 4.2.3 gives a complete list of reserved names and accounts; here we discuss only those which pertain directly to the protection of system software.

5.4.1 The root Person and the locksmith Group

The operating system provides a special super–user named **root**. The root person is the most powerful on the system. It overrides all protections in the ACL mechanism. The root person always has the UNIX user ID 0.

Use of root accounts should be carefully controlled. We suggest strongly that you assign passwords to root accounts, restrict knowledge of the passwords, and change the passwords periodically.

To use a root account, you can either log in as a root user or invoke the UNIX **su** command to switch your user ID to root.

Domain/OS also provides a super–user group named **locksmith**. Accounts of the form **%.locksmith.%** have the same overriding privileges as those of the form **root.%.%**. Of course, locksmith accounts warrant the same careful control. The locksmith group is not allowed to appear on project lists, so members of this group do not carry super–user privileges unless they explicitly log in with locksmith accounts.

5.4.2 The sys_admin, staff, and wheel Groups

Any operating system includes software whose use should be restricted. In Domain/OS, access to system software (both programs and data files) is restricted to persons such as **root** and **bin** or groups such as **sys_admin, staff,** or **wheel**. To protect your system software from corruption or deletion, you should create accounts with these persons and groups only for users who need to have access to system software.

5.4.3 The server Group

The **/etc/server** shell command and the **cps** Display Manager command create server processes and assign to them the SID **user.server.none**. Processes started via **server** or **cps** run independently of log–in activity.

When you log out, the DM does not terminate any process that run with server group identity. Therefore, we recommend that you do not create any accounts with the **server** group.

5.4.4 The user.none.none SID

The registry database contains a reserved account with the SID **user.none.none**. In the event that no registry servers are available, the operating system will allow anyone to log in under this SID without giving a password. This is the only SID under which you can log in when the registry is unavailable. Of course, when a registry server is available, the password is required.

You can use ACLs to limit the rights available to this SID.

5.5 Backups

Any user performing a backup must have the following access rights:

- Read access to all files and directories being backed up
- Write access to the **backup_history** file in the directory at the top of any tree to be backed up
- Execute (that is, search) access to all directories in the pathnames of objects being backed up

You can meet these requirements simply by performing all backups as **root**. If you prefer not to give root accounts to everyone who performs backups, we suggest the procedure below, which enables any **%.backup** account to run the backup utility with root privileges. The procedure creates a copy of **wbak** that runs with its user ID set to root and can be executed only by members of the group **%.backup**.

1. Use the **edrgy** command to create accounts with the group name **backup**.
2. Make a copy of the **wbak** utility. Set its protection to be setuid, owned by the **root** person and the **backup** group, and executable only by **root** and %.backup. (You must use a **root** or **%.locksmith** account to set the protection.)

```
% cd /usr/apollo/bin              (UNIX environments)
% cp wbak wbak.priv
% chown root wbak.priv
% chgrp backup wbak.priv
% chmod 4550 wbak.priv

% wd /usr/apollo/bin              (Aegis)
% cpf wbak wbak.priv
% edacl -p root prx -g backup rx -o none -ignore -w -none -setuid
  wbak.priv
```

You should perform Step 2 on every node that will run backups. Now the wbak.priv copies, executable only by **root** and **%.backup** accounts, will run as **root**. The original copies of **wbak** remain executable by any user, but without root privileges.

5.6 Protected Subsystems

A protected subsystem associates a set of data files (subsystem **data objects**) with programs (subsystem **managers**) that have special access rights to those files. *Using Your Aegis Environment* describes how to create protected subsystems. In this section, we discuss the login protected subsystem.

The following list shows the shell commands that are part of the login subsystem.

- The **login** command allows users to change their SIDs while logged in.
- The **/sys/siologin/siologin** command allows users to log in to a Domain node via a terminal or modem attached to a serial I/O (SIO) line.
- The **/sys/spm/spmlogin** command allows users to log in to one Domain node from another via the **crp -on** command.

Refer to the *Aegis Command Reference* for information about the shell commands listed above.

A user must "enter" a subsystem in order to set its attributes. Permission to enter a subsystem requires read and execute rights on the file **/sys/subsys/***subsystem*. Figure 5-9 shows the ACL for the **login** subsystem.

```
% acl /sys/subsys/login
Acl for /sys/subsys/login:
Subsystem login manager
Required entries
 root.%.%                          pr-x--
 %.sys_admin.%                     pr-x--
 %.%.none                          -r----
 %.%.%                             -r----
 Non-required entry rights mask:   ------
```

Figure 5-9. ACL for the login Subsystem

5.7 Control of Remote Access

The **lprotect** command and the **-lao** option to **edacl** allow you to control access from remote machines to objects on your machine.

The **node_owners** file in **'node_data** determines who can run **lprotect** on a node and also controls rights to perform other operations such as killing processes.

5.8 Installation and Protection

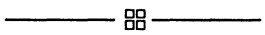
The **invol** system utility sets protection on some system software. Other system software is protected as it is installed. You should not alter this protection because the operating system depends on it. For example, some UNIX subsystems require that programs, data files, and directories be owned by particular users and groups or assigned particular permissions.

Note that protection model used and implemented by SR10 installation procedures depends on inheritance and the type of inheritance depends on how the disk was **invol**'d. You should be aware that, following installation, certain directories are not protected by inheritance, including **'node_data** and **/systest**.

Refer to *Installing Software with Apollo's Release and Installation Tools* for a description of how protection is inherited during installation. Refer to the BSD and SysV Command References for details on **invol**.

5.9 Registries and Protection

The registry database is protected in ways that do not use ACLs directly. Every entry in the registry database has an **owner** with rights to modify information pertaining to the entry. Owners of registry information are specified by SIDs; these can include wildcard (**%**) fields. See Chapter 4 for complete information about the registry.



Chapter 6

Line Printer Management

Contents

Chapter 6

Line Printer Management

This document describes the structure and installation procedure for the line printer spooling system developed for BSD.

The line printer system supports

- Multiple printers
- Multiple spooling queues
- Local and remote printers
- Printers attached via serial lines that require line initialization

The line printer system also supports raster output devices like Varian and Versatec* and laser printers like IMAGEN*.

The line printer system consists of the following files and commands:

/usr/lib/lpd	Line printer daemon
/usr/ucb/lpr	Program to enter a job in a printer queue
/usr/ucb/lpq	Spooling queue examination program
/usr/ucb/lprm	Program to delete jobs from a queue
/etc/lpc	Program to administer printers and spooling queues
/usr/spool/lpd/*	Spooling directories
/usr/spool/lpd/servername	Node on which **lpd** runs (optional)

*Versatec is a trademark of Versatec, Inc.; IMAGEN is a registered trademark of IMAGEN Corp.

You must log in as the super-user to run several of the components of the line printer system; that is, you must log in using the "root" user name and password.

6.1 How Does It Work?

Normally, only one node per Domain network will run the line printer daemon, **/usr/lib/lpd**. That node will usually start the daemon at boot time, by means of the **/etc/rc** file. When **/usr/lib/lpd** is started, the daemon goes through the **printcap** file and restarts any printers that have jobs in their queues. Then **lpd** listens for print requests from: nodes on your Domain network, nodes on another Domain network to which you're connected, and/or foreign hosts that are connected to your network.

When you submit a print request using the **lpr** command, **lpd** creates a copy of itself to process the request. (The original **lpd** process continues to listen for requests.) The **lpr** command places the actual material to be printed in the appropriate spool directory (**/usr/spool/lpd/***), and the copy of **lpd** then schedules the job's printing. If the printer you specified in the **lp** command line is unavailable for some reason, or if the machine to which it is connected is not operating, the request will remain in the spooling directory (or 'queue') until it is removed with the **lprm** command, or until the faulty printer or machine becomes available.

The file **/etc/printcap** is a database that describes the printers that are available to machines using the **lp** commands. The manual entry **printcap**(5) defines the format of this data base, as well as default values for important items like the directory in which spooling is performed.

> **NOTE:** At SR10, **/etc/printcap** is a per-node file. To use BSD print spooling, you should make sure that everyone has the same **printcap** file. We suggest that you have one **printcap** file that specifies all the printers (for example, **/usr/spool/lpd/etc.printcap**), and link all user **printcap** files to the common **printcap** file.

The Apollo version of the **printcap** file can have one additional entry, which is explained more fully in the *BSD Command Reference*. The additional entry is **pc**, which provides an interface to print commands you can use instead of sending output to **lp** or **rp**. In BSD, this is set to use the **prf** command sequence. If you want to communicate directly with the printer, set the **lp** field (device name) to the actual device name, and set the **if** field (input filter) to a filter appropriate for your application (for example, **/usr/lib/lpf**).

6.1.1 Prerequisites for BSD

From what we've said above, we can see that the **lp** system must have an **lpd** process running, and an entry in **/etc/printcap** for the printer to which requests will be sent. We'll assume that the necessary **lp** commands, like **lpc** and **lpr**, have been installed in the correct places on the system.

On Domain/OS, communication between a client (**lpr**, lpc, etc.) and the server (**lpd**) is accomplished by using the Network Compuing System (NCS). As a result, a local location broker (**/etc/llbd**) must be running on the node providing the line printer daemon. To make sure an **llbd** is running, create the file **/etc/daemons/llbd** before rebooting.

To set up **lpd** to run at boot time, uncomment (remove the # from the beginning of) the three lines in the **/etc/rc** file that read:

```
# if [ -f /usr/lib/lpd ]; then
# /usr/lib/lpd &
# fi
```

(Remember that the **/etc/daemons** directory must contain an empty file named **lpd** for the server to run.)

Shut down the node and restart it. The BSD implementation of **lpd** includes an optional file **/usr/spool/lpd/servername** that can be used if you want only one machine on your Domain network to run **lpd**. If the file exists, it must contain the TCP host name of the one machine on the network that is allowed to run **lpd**. If you attempt to start an **lpd** process on a machine other than the one specified in **/usr/spool/lpd/servername**, **lpd** will return an error message that specifies the name of the only machine that is allowed to run **lpd**. If the file does not exist, any number of machines on the network can run **lpd**, but only one should run it at a time.

The **/etc/printcap** file, as installed, contains several default descriptions for printer types. You may need to create an entry for another printer type. In this case, use the discussion of **printcap** later in this document, as well as the *BSD Command Reference* manual pages for **printcap**(5) and **termcap**(5), as your guide. Be certain not to leave a blank line between entries in the **/etc/printcap** file, as this will cause the **lp** system to think that there is a valid printer on the system with no name, and it may attempt to send requests there.

The **/etc/hosts.equiv** file is a list of machine names. In general, this file allows all the machines named in it to be treated as equivalent; for example, if your machine name is in this file, you can **rlogin** to any other machine named in the file, without going through the normal user and password authorization procedures. In order to use the line printer system on your network, a machine must have its TCP/IP host name in this file. There should be only one **/etc/hosts.equiv** file per network. (See *Configuring and Managing TCP/IP* for information on configuring TCP/IP correctly.

6.2 Commands

All of these commands, their options, and any arguments are explained fully on the *BSD Command Reference*.

6.2.1 Line Printer Daemon: lpd

The program **lpd**(8), usually invoked at boot time from the **/etc/rc** file, acts as a master server for coordinating and controlling the spooling queues configured in the **/etc/printcap** file. When **lpd** is started, it makes a single pass through the **/etc/printcap** database, restarting any printers which have jobs. In normal operation, **lpd** listens for service requests from other Apollo modes by implementing an NCS remote procedure call interface. Requests from foreign machines are handled on an Internet socket (under the "printer" service specification) for requests for printer access; see **socket**(2) and **services**(5) for more information on sockets and service specifications, respectively.

To provide service to foreign machines, a TCP server (**/etc/tcpd**)must be running on the same mode providing the **lpd** services. The **lpd** command spawns a copy of itself to process the request; the master daemon continues to listen for new requests.

Clients communicate with **lpd** using a simple transaction-oriented protocol. Remote clients are authenticated by means of the "privilege port" scheme employed by **rshd**(8C) and **rcmd**(3X). The following list shows the requests that **lpd** understands. In each request, the first byte indicates the "meaning" of the request, followed by the name of the printer to which it should be applied. Additional qualifiers may follow, depending on the request.

Request	Interpretation
^Aprinter\n	Check the queue for jobs and print any found
^Bprinter\n	Receive and queue job from another machine
^Cprinter [users] [jobs]\n	Return short list of current queue state
^Dprinter [users] [jobs]\n	Return long list of current queue state
^Eprinter person [users] [jobs]\n	Remove jobs from a queue

The **lpr**(1) command submits a print job to a local queue and notifies the local **lpd** that there are new jobs in the spooling area. The **lpd** command either schedules the job to be printed locally, or in the case of remote printing, attempts to forward the job to the appropriate machine. If the printer cannot be opened or if the destination machine is unreachable, the job will remain queued until it is possible to complete the work.

6.2.2 Show Line Printer Queue: lpq

The **lpq**(1) program works recursively backward, displaying the queue of the machine directly connected to the printer and then the queue(s) of the machine(s) that lead to it. The **lpq** has two forms of output. In the default short format, it gives a single line of

output per queued job. Figure 6-1 is an example of the short format. In the long format, it shows the list of files which comprise a job, and their sizes. Figure 6-2 is an example of the long format.

```
% lpq
 Rank    Owner                Job       Files                Total Size

 1st     cass                 18        memo, manual         4997 bytes
```

Figure 6-1. The lpq Short Format

```
% lpq -l
Warning: no daemon present

 cass: 1st                                        [job 018apollo]
       memo                                       456 bytes
       manual                                     4541 bytes
```

Figure 6-2. The lpq Long Format

If **lpr** is the last command in a pipeline, **lpq** cannot distinguish which files comprise the job. If you request the long format in this case, the legend "(standard input)" is displayed instead of the file names.

6.2.3 Remove Jobs from a Queue: lprm

The **lprm**(1) command deletes jobs from a spooling queue. If necessary, **lprm** will first kill off a running daemon that is servicing the queue, then restart it after the files are removed. When removing jobs destined for a remote printer, **lprm** acts like **lpq**, except that it first checks locally for jobs to remove and then tries to remove files in other queues off-machine. You must either be the owner of a job, or the super-user, to remove it.

6.2.4 Line Printer Control Program: lpc

The **lpc**(8) program controls the operation of the line printer system. You must log in as the super-user to use **lpc** and its associated commands. For each line printer configured in **/etc/printcap**, **lpc** can

- Disable or enable a printer
- Disable or enable a printer's spooling queue
- Rearrange the order of jobs in a spooling queue
- Find the status of printers, and their associated spooling Queues and printer daemons

6.3 Access Control

The printer system maintains protected spooling areas so that users cannot circumvent printer accounting or remove files other than their own. The following strategy is used to maintain protected spooling areas. The spooling area is writable only by a daemon user and daemon group. The **lpr** program runs setuid root and setgid daemon. The root access is used to read any file required, verifying accessibility with an **access**(2) call. The group ID is used in setting up proper ownership of files in the spooling area for **lprm**. Control files (**/usr/spool/lpd/*/cf***) in a spooling area are created by the **lpd** process, with daemon ownership and group ownership daemon. Their mode is 0660. This ensures that control files are not modified by a user and that no user can remove files except with **lprm**. The spooling programs, **lpd**, **lpq**, and **lprm** run **setuid** root and **setgid** daemon to access spool files and printers. The **lpd** process uses the same verification procedures as **rshd**(8C) in authenticating remote clients. As we mentioned earlier, the TCP host name of the node on which an **lp** user resides must be present in the file **/etc/hosts.equiv**.

6.4 Setting Up

The majority of the work in setting up is to create the **/etc/printcap** file and any printer filters for printers not supported in the distribution system. (The current BSD implementation of **lp** supports only the **lp** printer filter.)

6.4.1 Creating a printcap File

The **/etc/printcap** database contains one or more entries per printer. Each printer should have a separate spooling directory; otherwise, jobs will be printed on different printers, depending only on which printer daemon starts first. This section describes how to create entries for printers which do not conform to the default printer description.

Remote Printers

Printers that reside on remote hosts should have an empty **lp** entry. For example, the following **printcap** entry would send output to the printer named "lp" on the machine "vax".

```
lp|default line printer:\
:lp=:rm=vax:rp=lp:sd=/usr/spool/vaxlpd:
```

The **/etc/printcap** entry **rm** is the name of the remote machine to connect to; this name must appear in the **/etc/hosts** database, see **hosts**(5). The **rp** capability indicates that the name of the printer on the remote machine is "lp"; in this case, it could be left out, since this is the default value. The **sd** entry specifies **/usr/spool/vaxlpd** as the spooling directory instead of the default value of **/usr/spool/lpd/lp**.

A remote printer, for a machine on your Domain ring, is a printer on another Domain ring, or a printer on a foreign host to which your ring is connected via TCP/IP.

6.4.2 Output Filters

Filters are used to handle device dependencies and to perform accounting functions. The output filter **of** is used to filter text data to the printer device when accounting is not used or when all text data must be passed through a filter. It is not intended to perform accounting since it is started only once, all text files are filtered through it, and no provision is made for passing owner's log-in name, identifying the begining and ending of jobs, etc. The other filters (if specified) are started for each file printed and perform accounting if there is an **af** entry. If entries for both **of** and one of the other filters are specified, the output filter is used only to print the banner page; it is then stopped to allow other filters access to the printer.

6.5 Output Filter Specifications

The filters supplied with BSD handle printing and accounting for printers supported by the the **/com/prf** print command and **/com/prsvr** print server. For other printers or accounting methods, you may have to create a new filter.

Normally, **lpd** spawns filters, with standard input being the data to be printed, and standard output the printer. Standard error is attached to the **lf** file, for logging errors. A filter must return an exit code of 0 if there were no errors, 1 if the job should be reprinted, and 2 if the job should be discarded. When **lprm** sends a kill signal to the **lpd** process that is controlling printing, it sends a SIGINT signal to all filters and descendents of filters. If necessary, this signal can be trapped by filters that need to perform clean-up operations like deleting temporary files.

The arguments that may be passed to a filter depend on the filter's type. The **of** filter is called with the following arguments.

ofilter -w*width* -l*length*

The *width* and *length* values come from the **pw** and **pl** entries in the **/etc/printcap** database. The **if** filter is passed the following parameters:

filter [-c] -w*width* -l*length* -i*indent* -n *login* -h *host accounting_file*

The **-c** flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when the **-l** option of **lpr** is used to print the file). The **-w** and **-l** parameters are the same as for the **of** filter. The **-n** and **-h** parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from **/etc/printcap**.

All other filters are called with the following arguments:

filter -x*width* -y*length* -n *login* -h *host accounting_file*

The **-x** and -y options specify the horizontal and vertical page size in pixels (from the **px** and **py** entries in the **printcap** file). The rest of the arguments are the same as for the **if** filter.

Note that you can use **/usr/lib/lpf** as a generic means to filter data to an "unsmart" printer.

6.6 Line Printer Administration

The **lpc** program controls line printer activity. You must be logged in as the super-user to use **lpc**. The command format and other commands are described in lpc(8).

6.6.1 The abort and start Commands

The **abort** command terminates an active spooling daemon on the local host immediately and then disables printing (preventing new daemons from being started by **lpr**). This is normally used to forcibly restart a hung line printer daemon (that is, **lpq** reports that there is a daemon present but nothing is happening). It does not remove any jobs from the queue (use the **lprm** command instead). The **abort** command operates only on machines running the **lpd** process.

In addition, if you run **lpc** on a different node than the one that is running **lpd**, **abort** may kill the wrong process. If the node running **lpc** has a process with the same process ID as the process printing on the node running **lpd**, an **abort** will kill the first, or local, process, rather than the printing one.

The **start** command enables printing and requests **lpd** to start printing jobs.

6.6.2 The enable and disable Commands

The **enable** and **disable** commands allow spooling in the local queue to be turned on and off. This will allow or prevent **lpr** from putting new jobs in the spool queue. It is frequently convenient to turn spooling off while testing new line printer filters since the **root** user can still use **lpr** to put jobs in the queue, but no one else can. The other common use is to prevent users from putting jobs in the queue when the printer may be unavailable for a long time.

6.6.3 The restart Command

The **restart** command allows you to restart printer daemons when **lpq** reports that there is no daemon present.

6.6.4 The stop Command

The **stop** command is used to halt a spooling daemon after the current job completes; this also disables printing. This is a clean way to shut down a printer in order to perform maintenance, for example. Note that users can still enter jobs in a spool queue while a printer is stopped.

6.6.5 The topq Command

The **topq** command places jobs at the top of a printer queue. This can be used to reorder high priority jobs since **lpr** normally provides first-come-first-serve ordering of jobs.

6.7 Troubleshooting

There are a number of messages which may be generated by the the line printer system. This section categorizes the most common and explains the cause for their generation. Where the message indicates a failure, directions are given to remedy the problem.

In the examples below, *printer* is used in place of the actual name of the printer in the **/etc/printcap** database, and *host* is used as a substitute for the actual name of the host.

6.7.1 lpr Messages

`lpr: printer: unknown printer`

The *printer* was not found in the **/etc/printcap** database. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the **/etc/printcap** file.

`lpr: printer: jobs queued, but cannot start daemon.`

The connection to **lpd** on your Domain ring failed. This usually means the printer server started at boot time has died or is hung. Check the file **/usr/spool/lpd/servername** to see which machine should be running **lpd**, then verify that **lpd** is running on that machine, with the **ps**(1) command.

If the file does not exist, at least one TCP host on the ring must be running **lpd**. Use the command **/bin/hostname** to find out the TCP host name of your node, and make sure that the name is in the **/etc/hosts.equiv** file. Then start **/usr/lib/lpd** on your machine.

If the **ps** command shows **lpd** daemons, but they seem to be hung, do the following. Get a list of process identifiers of running **lpd**'s by typing

% **ps ax | fgrep lpd**

on the machine that is supposed to run **lpd**. The **lpd** to kill is the one that is not listed in any of the "lock" files. The lock file is contained in the spool directory of each printer (**/usr/spool/lpd/***). Kill the master daemon by using the following command:

% **kill** *pid*

where *pid* is the process ID number of the **lpd** process, as reported by the **ps** command. Then restart the daemon (and printer) with the following command:

% **/usr/lib/lpd**

Another possibility is that the **lpr** program is not **setuid** *root*, **setgid** *daemon*. This can be checked with

% **ls -lg /bin/lpr**

`lpr: printer: printer queue is disabled`

This means the queue was turned off with

% **lpc disable printer**

to prevent **lpr** from putting files in the queue. This is normally done by the system manager when a printer is going to be down for a long time. The printer can be turned back on by a super-user with **lpc**.

6.7.2 lpq Messages

`waiting for` *printer* `to become ready (offline ?)`

The printer device could not be opened by the daemon. This can happen for a number of reasons, the most common being that the printer is turned offline. This message can also be generated if the printer is out of paper, the paper is jammed, etc. The actual reason depends on the meaning of error codes returned by system device driver. Not all printers supply sufficient information to distinguish when a printer is offline or having trouble (for example, a printer connected through a serial line). Another possible cause of this message is that some other process, such as an output filter, has an exclusive open on the device. Your only recourse here is to kill off the offending program(s) and restart the printer with **lpc**.

printer `is ready and printing`

The **lpq** program checks to see if a daemon process exists for *printer* and prints the file status. If the daemon is hung, a super-user can use **lpc** to abort the current daemon and start a new one.

`waiting for` *host* `to come up`

This indicates there is a daemon trying to connect to the remote machine named *host* in order to send the files in the local queue. If the remote machine is up, **lpd** on the remote machine is probably dead or hung and should be restarted as mentioned for **lpr**.

`sending to` *host*

The files should be in the process of being transferred to the remote host. If not, the local daemon should be aborted and started with **lpc**.

`Warning:` *printer* `is down`

The *printer* has been marked with **lpc** as being unavailable.

`Warning: no daemon present`

The **lpd** process overseeing the spooling queue, as indicated in the "lock" file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly died. The error log file for the printer should be checked for a diagnostic from the deceased process. To restart an **lpd**, use

```
% lpc restart printer
```

This error might also be reported if **lpq** is not run on the same machine as **lpd**.

6.7.3 lprm Messages

`lprm: printer: cannot restart printer daemon`

This case is the same as when **lpr** prints that the daemon cannot be started.

6.7.4 lpd Messages

The **lpd** program can write many different messages to the error log file (the file specified in the **lf** entry in **/etc/printcap**). Most of these messages are about files which can not be opened and usually implicate the **/etc/printcap** file or imply that the protection modes of the files are not correct. Files may also be inaccessible if people manually manipulate the line printer system (that is, bypass the **lpr** program).

In addition to messages generated by **lpd**, any of the filters that **lpd** spawns may also log messages to this file.

6.7.5 lpc Messages

`couldn't start printer`

This case is the same as when **lpr** reports that the daemon cannot be started.

`cannot examine spool directory`

Error messages beginning with "couldn't ..." are usually due to incorrect ownership and/or protection mode of the lock file, spooling directory or the **lpc** program.

6.7.6 General TCP/IP Error Conditions

`Socket: I/O Error`

This is a general indication of problems with the TCP connection. Restarting **tcpd**, on either your machine or the one running **lpd** (if they are different), or both, is the first thing you should try. If this doesn't work, check that any TCP/IP links you've set up are pointing to the correct place, and that all the machines that must communicate are up.

Chapter 7

uucp Administration

Contents

Chapter 7

uucp Administration

The **uucp** program is the central program in a group of programs that, together, permit communication between Domain systems by using either dial-up or hard-wired connections. SR10 supplies the HoneyDanBer version of uucp. This version provides interoperability between SR9.7 and SR10 nodes and communicates with most versions of **uucp**, including Apollo pre-SR10 nodes, other HoneyDanBer **uucp** implementations and other **uucp** implementations (it does not support communications over TCP, TLI or X.25). This Apollo implementation of uucp also provides enhancements such as spool directory trees, better security, and error handling.

This chapter describes how a **uucp** network is set up, the format of control files, and administrative procedures. You should be familiar with the manual pages for each of the **uucp** related commands. Other helpful external documentation on HoneyDanBer **uucp** include the following:

- AT&T *UNIX System V User's Guide*
- AT&T *UNIX System V System Administrator's Guide*
- *UNIX System Security*, Patrick H. Wood and Stephen G. Kochan, Hayden Books
- *HoneyDanBer uucp — Bringing UNIX Systems into the Information Age*, Bill Rieken and Jim Webb

Refer to *Making the Transition to SR10 Operating System Releases* for more information on **uucp** interoperability in mixed networks (SR9.7 and SR10 nodes).

7.1 Network Planning

There are several considerations you should take into account before configuring a network. These considerations are discussed below.

7.1.1 Extent of the Network

Before you set up the network configuration, you must make some basic decisions about access to processors in the network. If you control only one processor and are joining an existing network, then you must decide what level of access you will grant to other systems. Likewise, the other members of the network must make a similar decision for the system you control.

You control system access by using the following:

- The UNIX system **passwd** command to grant access to systems.
- The file **/usr/lib/uucp/Permissions** to restrict access to specified parts of the file system tree.
- The file **/usr/lib/uucp/Systems** on the local processor to determine how many other systems on the network you can reach.

If you are setting up more than one processor, and thus controlling a larger portion of the network, you must make more decisions about the setup. For example, the network can be set up as a private network where only those machines under the direct control of the administrator can access each other. Granting no access to machines outside the network can be done if security is paramount; however, this is usually impractical. Very limited access can be granted to outside machines by each of the systems on the private network. Alternatively, access to/from the outside world can be confined to only one processor. This is frequently done to minimize the effort in keeping access information (passwords, phone numbers, log-in sequences, etc.) updated and to minimize the number of security holes for the private network.

7.1.2 Hardware and Line Speeds

The **uucp** system supports two means of interconnection:

- Direct connection using a null modem cable
- Connection over the Direct Distance Dialing (DDD) network

These methods of interconnection support the following dialers and networks:

- Intelligent auto-dial modems
- Dialers attached to Local Area Networks (LANs)

In choosing hardware, the equipment used by other processors on the network must be considered. For example, if some systems on the network have only 103-type (300-baud) data sets, then communication with them is not possible unless the local system has a 300-baud data set connected to a calling unit. (Most data sets available on systems are 1200-baud.)

If hard-wired connections are to be used between systems, then the distance between systems must be considered since a null modem cannot be used when the systems are separated by more than several hundred feet. The limit for communication at 9600-baud is about 800 to 1000 feet. However, the RS-232 specification and Western Electric Support Groups only allow for less than 50 feet. Limited-distance modems must be used beyond 50 feet as noise on the lines becomes a problem.

NOTE: Ensure that you have **getty** or **siologin** monitoring the port to which **uucp** is connected.

7.1.3 Maintenance and Administration

There is a minimum amount of maintenance that must be provided in each system to keep the access files updated, to ensure that the network is running properly, and to track down line problems. When more than one system is involved, the job becomes more difficult because there are more files to update and because users are much less patient when failures occur between machines that are under local control.

7.2 The uucp Software

To call another system, the **uucp**(1) or **uux**(1) command queues users requests and spawns the **uucico** daemon. The **uucico** daemon initiates the call to another system and performs the file transfer. On the receiving side, **uucico** is invoked to receive the transfer. Remote execution jobs are actually done by transferring a command file to the remote system and invoking a daemon (**uuxqt**) to execute that command file and return the results.

7.3 Installation

The **uucp**(1) package is installed as part of the standard UNIX environment.

The following object modules are installed as part of standard installation:

- **uucp** — The file transfer command.
- **uux** — The remote execution command.
- **uucico** — The **uucp** network daemon.
- **uustat** — Network status command.
- **uucleanup** — Clean-up command.
- **uuxqt** — The remote execution daemon.
- **uucheck** — The command that checks for the presence of the **uucp** system-required files and directories.
- **uuname** — The command to list the names of systems known to **uucp**.
- **uusched** — The **uucp** file transport scheduler.
- **uuencode** — The **uucp** command that sends binary files to a remote system via mail.
- **uudecode** — The **uucp** command that recreates the original file with the specified mode and name.

NOTE: **uuencode** does not preserve Aegis file type information. The **uudecode** command always added produces files of type unstruct. Therefore, to transfer an obj type file, use the **obty** command to change the type to unstruct. Use **uuencode** on the file, tranfer it to the destination system, and **uudecode** it. Use the **obty** command to modify the type to obj.

The following shell scripts are installed as part of normal installation:

- **SetUp** — Sets up all necessary **uucp** system files.
- **Uutry** — Starts a **uucico** for the specified system.
- **remote.unknown** — Logs calls from unknown systems.
- **uudemon.admin** — Sends **uucp** status information to the system administrator node.
- **uudemon.cleanup** — Cleans up **uucp** directories.
- **uudemon.hour** — Starts **uusched** and **uuxgt**.
- **uudemon.poll** — Sets up for polling systems.
- **Maxuuxgts** — Defines the maximum number of **uuxgt** programs that can run at one time.
- **Maxuuscheds** — Defines the maximum number of **unsched** programs that can run at one time.
- **uulog** — Prints log information for the specified system.
- **uuto** and **uupick** — Public UNIX system to UNIX system file copy.

The **uucp** programs look for the file, **/usr/lib/uucp/sitename**, to determine the **uucp** site name. This is an ASCII file containing the **uucp** site name and must be created by the system administrator. If this file does not exist, the site name is set to the node name returned by the **gethostname** call.

7.3.1 Apollo Configuration

In a Domain network, there is usually one node configured as the **uucp** "site." Other nodes on the network are set up so that their **/usr/spool/uucp** and **/usr/lib/uucp** directories link to the **uucp** site.

The **uucp** and **uux** programs perform file permission checking and queue requests to the **/usr/spool/uucp** directory for each node in the network. When the **uucico** program starts on the **uucp** site node, it looks in the **/usr/spool/uucp** directories for requests to perform.

7.3.2 Differences between /dev/siox and /dev/ttyx Devices

The tty/sio ports used by **uucp** must be configured to be compatible with the attached modem (for example, baud rate, bits per character, parity). Although **/dev/sio***x* and **/dev/tty***x* (where *x* is the port number) refer to the same physical port, they are handled differently. Operations on **/dev/sio***x* ignore DCD (Data Carrier Detect) status while default operations on **/dev/tty***x* adhere to the UNIX conventions (such as requiring DCD to be high before completing an open of the device).

The **uucp** programs open the device with the O_NDELAY flag, which causes the open to return without waiting for DCD. This enables either **/dev/sio***x* or **/dev/tty***x* devices to be used for both dialing out or receiving calls. However, it is recommended that **/dev/tty***x* devices be used exclusively for **uucp** (including **cu** and **tip**). This enables device locking to function properly.

> **NOTE:** To monitor the tty/sio port for incoming calls, add and entry to the **/etc/ttys** file to start **/etc/getty**. It is recommended for the reasons stated above, that **/dev/tty***x* be used for **getty**. A port monitored by **/etc/getty** cannot be used to dial out on unless **getty** is suspended first. This can be done by using a script.

7.3.3 Password File

The registry contains the information necessary to generate the **/etc/passwd** file that contains password entries for remote **uucp** logins. These entries allow remote systems to call the local system. For example:

```
nuucp:zaaAA:6:1:uucp.Admin:/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

The **uucico** daemon is specified for the shell field, and the **spool** directory is specified as the working directory. There must also be an entry a **uucp** administrative login. This login is the owner of all the **uucp** object and spooled data files and is usually named **uucp**. For example, the following is a entry in **/etc/passwd** for this administrative login:

```
uucp:zAvLCKp:5:1:uucp.Admin:/usr/lib/uucp:
```

Note that the standard shell is used instead of **uucico**.

The log–in account for sites calling into the **uucp** site must have its shell defined to be **/usr/lib/uucp/uucico.real**.

7.4 Supporting Database

The following Basic Networking Utilities support files are in the **/usr/lib/uucp** directory:

- **Devices** — Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices.
- **Dialers** — Contains character strings required to negotiate with network devices (automatic calling devices) to establish connections to remote computers (non 801-type dialers).
- **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.
- **Dialcodes** — Contains dial-code abbreviations that may be used in the phone number field of **Systems** file entries.
- **Permissions** — Defines the level of access granted to computers when they attempt to transfer files or remotely execute commands on your computer.
- **Poll** — Defines computers that are to be polled by your system and when they are polled.
- **Devconfig** — Used to configure utilities for the Basic Networking Utilities on a STARLAN NETWORK or some other transport provider that conforms to the AT&T Transport Interface.
- **Sysfiles** — Assigns different or multiple files to be used by **uucico** and **cu** as **Systems**, **Devices**, and **Dialers** files.

There are several other files that may be considered part of the supporting data base, but are not directly related to the process of establishing a link and transferring files. These files (**Maxuuxqts**, **Maxuuscheds**, and **remote.unknown**) are described in Section 7.3, "Installation."

Each of the supporting data base files is described below.

7.4.1 Devices File

The **Devices** file (**/usr/lib/uucp/Devices**) contains information for all devices used to establish a link to a remote computer, devices such as automatic call units, direct links, and network connections.

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

The meaning of each of these fields is:

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 (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: nuucp \ssword: Oakgrass`

  You can designate a protocol to use for a device within this field. See the "Protocols" section at the end of the description of this file.

Line This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the Automatic Dial Modem 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** was used 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 dash (-) 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 follows:

Devices: `ACU tty11 - D1200 penril`

Systems: `eagle Any ACU D1200 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.

Dia*ler-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 one of several special dialer types. The following special dialer types are compiled into the software and are therefore available without having entries in the **Dialers** file:

801 — Bell 801 auto dialer

TLI — Transport Level Interface Network (without STREAMS)

TLIS — Transport Level Interface Network (with STREAMS)

The *Dialer–Token–Pairs* (*DTP*) field may be structured in the following four ways, depending on the device associated with the entry:

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 ttyl1 - 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. (**\T** is implied, see below.) Backslash sequences are described below.

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 ttyl3 - 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 usingthe **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).

Protocols

You can define the protocol to use with each device. In most cases it is not needed since you can use the default or define the protocol with the particular system you are calling (see **Systems** file, *Type* field). If you do specify the protocol, you must do in the form *Type,Protocol*. Currently, the only available protocol is

g This protocol is slower and more reliable than **e**. It is good for transmission over noisy telephone lines.

7.4.2 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, and it is often used to dial a phone number using an ASCII dialer (such as the Automatic Dial Modem).

As shown previously, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801, TLI, or TLIS). 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 starting with the seventh position is used as an index into the **Dialers** file. If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations. Each entry in the **Dialers** file has the following format:

dialer substitutions expect-send ...

The **Dialer** field matches the fifth and additional odd numbered fields in the **Devices** file. The **Substitutions** field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate the equal sign (=) and the dash (-) 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 Basic Networking Utilities in the **Dialers** file:

```
penril =W-P "" \d > s\p9\c)-W\p\r\ds\9\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
rixon =&-% "" \d\r\r\c $ s9\c)-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadiac =K-K "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
develcon "" "" \pr\ps\c est:\007 \E\D\e \007
micom "" "" \s\c NAME? \D\r\c GO
direct
att2212c  =+-, "" \r\c :--: atol2=y,T\T\r\c red
att4000 =,-, "" \033\r\r\c DEM: \033s0401\c \006 \033s0901\c \
  \006 \033s1001\c \006 \033s1102\c \006 \033dT\T\r\c \006
att2224 =+-, "" \r\c :--: T\T\r\c red
nls  "" "" NLPS:000:001:1\N\c
```

There are also three AT&T modems that have entries in the **Dialers** file. The meaning of some of the escape characters (those beginning with "\") used in the **Dialers** file are listed below:

\p	Pause (approximately 12 to 14 seconds).
\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 newline or carriage return.
\n	Send newline.
\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 greater than symbol (>).
s\p9\c	Send an "s", pause for 1/2 second, send a "9", send no terminating newline.
)-W\p\r\ds\p9\c-)	Wait for a ")" character. If it is not received, process the string between the "-" characters as follows. Send a "W", pause, send a carriage-return, delay, send an "s", pause, send a "9", without a newline, and then wait for the ")" character.
y\c	Send a "y".
:	Wait for a colon (:).
\E\TP	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 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. Then send a "P".
>	Wait for a greater-than sign (>).
9\c	Send a "9" without a newline.
OK	Waiting for the string "OK".

7.4.3 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, the basic networking 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 sequential order.

Using the **Sysfiles**, you can define several files to be used as "Systems" files. See Subsection 7.4.8 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 below.

System-name This field contains the node name of the remote computer.

Time This field is a string that indicates the day and time when the remote computer can be called. The format of the *Time* field is:

daytime [;retry]

The *day* portion may be a list containing:

Su Mo Tu We Th Fr Sa for individual days

Wk For any weekday (Mo Tu We Th Fr)

Any For any day

Never For a passive arrangement with the remote computer. 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 in respect to the remote computer (see Subsection 7.4.5 for a discussion of the **Permissions** file).

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.

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 Thursday, 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.

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 \`

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 dash (-) 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 (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 \`
`assword: 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 log-in 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 nothing followed by a newline, 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 newline 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: xyzzy
```

This example says send a carriage return and wait for **ogin:** (for Login:). If you don't get **ogin:**, send a BREAK. When you do get **ogin:** send the log-in name **uucpx**, then when you get **word:** (for Password:), send the password **xyzzy**.

There are several escape characters that cause specific actions when they are a part of a string sent during the log–in 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 2 seconds before sending or reading more characters.
\p	Pause for approximately 12 to 14 seconds.
\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 newline 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 backslash (\) character.
EOT	Send or expect EOT newline twice.
BREAK	Send or expect a break character.
\K	Same as BREAK.
\ddd	Collapse the octal digits (ddd) into a single character.

Some versions of **uucp**, including the SR9.7 Domain/IX™ version, send the log–in sequence with even parity by default. The SR10 log–in program expects no parity. To change the parity of the log–in sequence sent, add the expect/send sequence of "" **P_ZERO** before the log–in expect/send sequence in the **L.sys file.** for example.

7.4.4 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**.

7.4.5 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.

How Entries are Structured

Each entry is a logical line with physical lines terminated by a backslash (\)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 pound sign (#) and they occupy the entire line up to a newline character. Blank lines are ignored (even within multiline 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.

Considerations

The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote computers:

- All log–in IDs used by remote computers to log in for **uucp** communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry will have the following default permissions/restrictions:
  - Local send and receive requests will be executed.
  - The remote computer can send files to your computer's **/usr/spool/uucppublic** directory.
  - The commands sent by the remote computer for execution on your computer must be one of the default commands, usually **rmail**.

Options

This subsection describes each option, tells how it is used, and lists its 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.

`REQUEST=yes`
Specifies that the remote computer can request to transfer files from your computer.

`REQUEST=no`
specifies that the remote computer cannot request to receive files fromyour 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 SENDFILE 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=/usr/spool/uucppublic
WRITE=/usr/spool/uucppublic
```

The strings

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a colon separated list of pathnames. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full pathname of a file coming in or going out. To grant permission to deposit files in **/usr/news** as well as the public directory, the following values would be used with the WRITE option:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

It should be pointed out that if the READ and WRITE options are used, all pathnames must be specified because the pathnames are not added to the default list. For instance, if the **/usr/news** pathname was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote 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=/usr/spool/uucppublic
```

would permit reading any file except those in the **/etc** directory (and its subdirectories) and writing only to the default **/usr/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:rnews:lp
```

overrides the COMMAND default so that the computers **owl, raven, hawk,** and **dove** can now execute **rmail, rnews,** and **lp** on your computer.

In addition to the names as specified above, there can be full pathnames of commands. For example,

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp
```

specifies that command **rmail** uses the default path. The default paths for your computer are **/bin, /usr/bin,** and **/usr/lbin**. When the remote computer specifies **rnews** or **/usr/lbin/rnews** for the command to be executed, **/usr/lbin/rnews** will be executed regardless of the default path. Likewise, **/usr/local/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/lbin/rnews:ALL:/usr/local/lp
```

illustrates two points: The **ALL** value can appear anywhere in the string, and the pathnames specified for **rnews** and **lp** will be used (instead of the default) if the requested command does not contain the full pathnames for **rnews** 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; although, 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. Therefore, if you cannot trust someone on the remote computer, do not provide that computer with a privileged login and password.

The following 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/lbin/rnews \
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/lbin/rnews** 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.

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:rnews:/usr/lbin/Photo:/usr/lbin/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=/
```

7.4.6 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 finally 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**.

7.4.7 Devconfig File

The **/usr/lib/uucp/Devconfig** file is used when your computer communicates over a STARLAN network or some other Streams-based transport provider that conforms to the AT&T Transport Interface (TI). Note that **uucp** over TI is not supported with this release.

Devconfig entries define the STREAMS modules that are used for a particular TI device. Entries in the **Devconfig** file have the format:

service=x device=y push=z [:z ...]

where *x* can be **cu**, **uucico**, or both separated by a colon; *y* is the name of a TI network and must match an entry in the **Devices** file; and *z* is replaced by the names of STREAMS modules in the order that they are to be pushed onto the Stream. Different modules and devices can be defined for **cu** and **uucp** services.

The following entries should most commonly be used in the file:

```
service=cu  device=STARLAN push=ntty:tirdwr:ld0
service=uucico device=STARLAN push=ntty:tirdwr:ld0
```

This example pushes **ntty** then **tirdwr**, then **ld0**.

7.4.8 Sysfiles File

The **/usr/lib/uucp/Sysfiles** file lets you assign different files to be used by **uucp** and **cu** as **Systems**, **Devices**, and **Dialers** files. This optional file may be useful in the following cases:

- You may want different **Systems** files so requests for log in 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.

7.5 Administration

The role of the **uucp** administrator depends heavily on the amount of traffic that enters or leaves a system and the quality of the connections that can be made to and from that system. For the average system, only a modest amount of traffic (100 to 200 files per day) pass through the system and little if any intervention with the **uucp** automatic cleanup functions is necessary. Systems that pass large numbers of files (200 to 10,000) may require more attention when problems occur.

The following subsections describe the routine administrative tasks that must be performed by the administrator or are automatically performed by the **uucp** package. A subsection on problems describes frequent problems and how to deal with them effectively.

7.5.1 Cleanup

The biggest problem in a dial-up network like **uucp** is dealing with the backlog of jobs that cannot be transmitted to other systems. The following clean-up activities should be routinely performed by shell scripts started from **cron**(1).

Cleanup of Undeliverable Jobs

The **uudemon.cleanup** script usually contains an invocation of the **uucleanup** command to purge any jobs that are older than some fixed time (usually 72 hours). It is also used to purge any **lock** or **status** files. An sample invocation of **uucleanup**(1M) to remove old work, data, and execute files follows:

```
/usr/lib/uucp/uucleanup -c7 -d7 -x7 -w7
```

7.5.2 Polling Other Systems

Systems that are passive members of the network must be polled by other systems in order for their files to be sent. To do this, set up the **Poll** file (**/usr/lib/uucp/poll**) with entries in the following format:

sys-name hour . . .

When the **uudemon.poll** script is run, it sets up the approproiate work files in the **/spool** directory. These will be seen and processed by the scheduler.

7.5.3 Problems

The following subsections list the most frequent problems that appear on systems that make heavy use of **uucp**.

Out of Space

The file system used to spool incoming or outgoing jobs can run out of space and prevent jobs from being spawned or received from remote systems. The inability to receive jobs is the worse of the two conditions. When file space does become available, the system will be flooded with the backlog of traffic.

Bad ACU and Modems

The ACU and incoming modems occasionally cause problems that make it difficult to contact other systems or to receive files. These problems are usually readily identifiable since files in the **/.Log** directory will usually have entries that point to the bad line. If a bad line is suspected, use the **cu** command to try calling another system using the suspected line.

Administrative Problems

Some **uucp** networks have so many members that it is difficult to keep track of changing passwords, changing phone numbers, or changing logins on remote systems. This can be a very costly problem since ACUs will be tied up calling a system that cannot be reached.

7.6 Debugging

In order to verify that a system on the network can be contacted, the **uucico** daemon can be invoked directly from a user's terminal directly. For example, to verify that **mhtsd** can be contacted, a job would be queued for that system as follows:

```
uucp -r file mhtsd!~/tom
```

The **-r** option forces the job to be queued but does not invoke the daemon to process the job. The **uucico** command can then be invoked directly:

```
/usr/lib/uucp/uucico -r1 -x4 -smhtsd
```

The **-r1** option is necessary to indicate that the daemon is to start up in **master** mode (i.e., it is the calling system). The **-x4** specifies the level of debugging that is to be printed. Higher levels of debugging can be printed (greater than 4) but require familiarity with the internals of **uucico**.

If several jobs are queued for the remote system, it is not possible to force **uucico** to send one particular job first.

The contents of the files in the **//usr/spool/uucp/.Log** directory should also be monitored for any error indications that are posted in that directory. Frequently, problems can be isolated by examining the entries in the file corresponding to a particular system. For example, to examine log files for a system named "apollo," you would look at **//usr/spool/uucp/.Log/uucp/apollo**. This structure is illustrated in Figure 7-1.

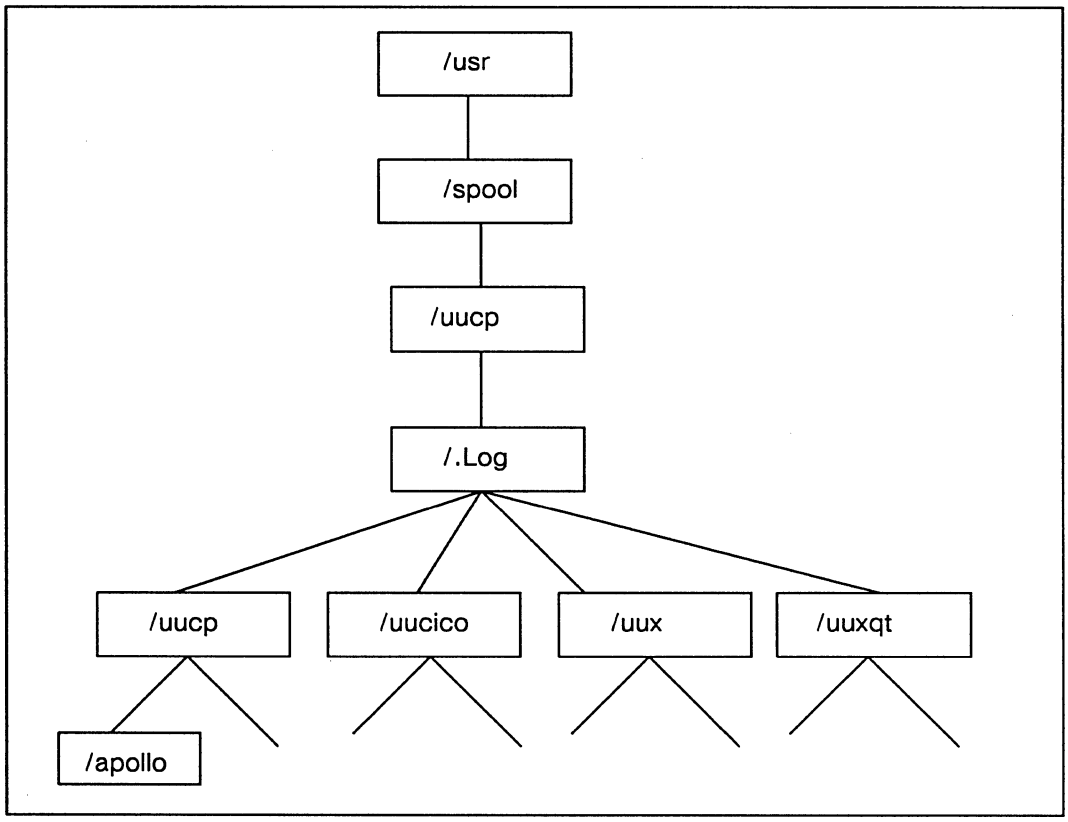


Figure 7–1. Directory Structure

Chapter 8

sendmail Configuration and Usage

Contents

Chapter 8

sendmail Configuration and Usage

Routing mail through a heterogeneous internet presents many problems. Among the worst of these is that of address mapping. Historically, this has been handled on an ad hoc basis. However, this approach has become unmanageable as internets grow.

The **sendmail** facility acts a unified "post office" to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both domain-based addressing and old-style ad hoc addresses. The production system is powerful enough to rewrite addresses in the message header to conform to the standards of a number of common target networks, including old ARPANET, new ARPANET, **uucp**, and Phonenet. The **sendmail** facility also implements a Simple Mail Transfer Protocol (SMTP) server, message queueing, and aliasing.

NOTE: This chapter is largely taken from the original Berkeley *sendmail Installation and Operation Guide*, and the paper *Introduction to sendmail* by Eric Allman.

8.1 Introduction

The **sendmail** facility implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair; in particular, the mail system can refer to users using a host-name-user-name pair. Host names and numbers have to be administered by a central authority, but user names can be assigned locally to each host.

In an internet, multiple networks with different characteristics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled simply by ad hoc techniques, such as providing network names that appear local to hosts on other networks, as with the ETHERNET* at Xerox-PARC. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem since only

*ETHERNET is a registered trademark of Xerox Corporation.

adjacent hosts must be entered into the system tables, while others use end–to–end addressing. Some networks use a left–associative syntax and others use a right–associative syntax, causing ambiguity in mixed addresses.

The **sendmail** facility is intended to help bridge the gap between the totally ad hoc world of networks that know nothing of each other and the clean, tightly coupled world of unique network numbers. It can accept arbitrary address syntaxes and domain–based addressing, resolving ambiguities by using heuristics specified by the system administrator. It helps guide the conversion of message formats between disparate networks. In short, **sendmail** is designed to assist a graceful transition to consistent internetwork addressing schemes.

8.2 Interfaces to the Outside World

There are three ways **sendmail** can communicate with the outside world, both in receiving and in sending mail. The **sendmail** utility communicates by using the conventional UNIX argument vector/return status, speaking SMTP over a pair of UNIX pipes, and speaking SMTP over an interprocess (or) channel.

8.2.1 Argument Vector/Exit Status

This technique is the standard UNIX method for communicating with the process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

8.2.2 SMTP Over Pipes

The SMTP protocol can be used to run an interactive lock–step interface with the mailer. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes standard input. Anything appearing on the standard output must be a reply code in a special format.

8.2.3 SMTP Over an IPC Connection

This technique is similar to the previous technique, except that it uses an IPC channel. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a **sendmail** process on another machine.

8.3 Configuration

Almost all configuration information is read at run time from an ASCII file: encoding macro definitions (the value of macros used internally), header declarations (the format of header lines that sendmail will process specially,) mailer definitions (information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which used to parse and rewrite the addresses).

Configuration is controlled primarily by a configuration file read at startup. If the mail is being sent by a local user, and the file ".mailcf" exists in the sender's home directory, that file is read as a configuration file after the system configuration file. The primary use of this feature is to add header lines.

The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options. Refer to Section 8.8 for details on the configuration file.

Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name **sendmail** will use to identify itself in error messages. Other macros transmit information from **sendmail** to the configuration file for use in creating other fields (such as argument vectors to mailers); for example, the name of the sender, and the host and user of the recipient. Other macros are unused internally, and can be used as shorthand in the configuration file.

Header Declarations

Header declarations inform **sendmail** of the format of known header lines. Knowledge of a few header lines is built into **sendmail**, such as the "From:" and "Date:" lines.

Most configured headers will be automatically inserted in the outgoing message if they don't exist in the incoming message. Certain headers are suppressed by some mailers.

Mailer Declarations

Mailer declarations tell **sendmail** of the various mailers available to it. The definition specifies the internal name of the mailer, the pathname of the program to call, some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

Address Rewriting Rules

The configuration file supports the editing of addresses into different formats. For example, an address of the form:

ucsfcgl!tef

might be mapped into the following to conform to the Domain/OS syntax:

tef@ucsfcgl.uucp

Translations can also be done in the other direction.

8.4 Installation

Due to the requirements of flexibility for **sendmail**, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Note that Apollo's pre-installed software enables both SysV and Berkeley mailers to use **sendmail** as is.

The remainder of this chapter assumes that you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the **sendmail** subtree.

8.4.1 Off-the-Shelf Configurations

Apollo UNIX environments are supplied with three sample configuration files. These files are extensively commented and can work "as is" for many sites.

- **/usr/lib/uucpproto.cf**
- **/usr/lib/arpaproto.cf**
- **/usr/lib/sendmail.cf**

These off-the-shelf configuration files are supplied to handle the basic cases: **/usr/lib/arpaproto.cf** for ARPANET (TCP) sites and **/usr/lib/uucpproto.cf** for **uucp** sites. To install **uucpproto.cf** as your configuration, for example, you could use the following commands:

```
% cd /usr/lib

% cp uucpproto.cf sendmail.cf
```

The off-the-shelf configuration files will get you started with **sendmail**. However, you can have to adjust the configuration file to meet the needs of your particular mail delivery system.

8.5 sendmail Arguments, Configuration Options, and Mailer Flags

The **sendmail** command, **/usr/lib/sendmail**, provides a set of arguments and options that enable you to specify the kind of processing you want. This section describes the arguments and options in detail. Arguments must be presented with flags before addresses. The flags are

-f *addr* — The sender's machine address is *addr*. This flag is ignored unless the real user is listed as a "trusted user" or if *addr* contains an exclamation point (because of certain restrictions in **uucp**).

-r *addr* — An obsolete form of **-f**.

-h*cnt* — Sets the "hop count" to *cnt*. This represents the number of times this message has been processed by **sendmail** (to the extent that it is supported by the underlying networks).The *cnt* is incremented during processing, and if it reaches MAXHOP (currently 30) **sendmail** throws away the message with an error.

-F*name* — Sets the full name of this user to *name*.

-n Don't do aliasing or forwarding.

-t Read the header for "To:", "Cc:", and "Bcc:" lines, and send to everyone listed in those lists. The "Bcc:" line will be deleted before sending. Any addresses in the argument vector will be deleted from the send list.

-b*x* Set operation mode to *x*. Operation modes are

m	Deliver mail (default)
a	Run in arpanet mode (see below)
s	Speak SMTP on input side
d	Run as a daemon
t	Run in test mode
v	Just verify addresses, don't collect or deliver
i	Initialize the alias database
p	Print the mail queue
z	Freeze the configuration file (Not currently supported)

The special processing for the ARPANET includes reading the "From:" line from the header to find the sender, printing ARPANET style messages (preceded by three–digit reply codes for compatibility with the FTP protocol), and ending lines of error messages with <CRLF>.

-q*time* Try to process the queued up mail. If the time is given, a **sendmail** will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.

-C*file* Use a different configuration file.

-d*level* Set debugging level.

-o*xvalue* Set option *x* to the specified *value*. These options are described in the below.

Configuration Options

The following options can be set using the **-o** flag on the command line or the **O** line in the configuration file:

A*file* Use the named *file* as the alias file. If no *file* is specified, use aliases in the current directory.

a If set, wait for an "@:@" entry to exist in the alias database before starting up. If it does not appear in five minutes, rebuild the database.

c If an outgoing mailer is marked as being expensive, don't connect immediately. This requires that queueing be compiled in, since it will depend on a queue run process to actually send the mail.

d*x* Deliver in mode *x*. Legal modes are

i	Deliver interactively (synchronously)
b	Deliver in background (asynchronously)
q	Just queue the message (deliver during queue run)

D If set, rebuild the alias database if necessary and possible. If this option is not set, **sendmail** will never rebuild the alias database unless explicitly requested using **-bi**.

e*x* Dispose of errors using mode *x*. The values for *x* are:

p	Print error messages (default)
q	No messages, just give exit status
m	Mail back errors
w	Write back errors (mail if user not logged in)
e	Mail back errors and give 0 exit stat always

F*n* The temporary file mode, in octal. 644 and 600 are good choices.

f Save the UNIX system "From" lines at the front of headers. Normally they are assumed redundant and discarded.

g*n* Set the default group ID for mailers to run in to *n*.

Hfile Specify the help file for SMTP.

i Ignore dots in incoming messages.

L*n* Set the default log level to *n*.

M*xvalue* Set the macro *x* to *value*. This is intended only for use from the command line.

m Send to me too, even if I am in an alias expansion.

o Assume that the headers contain spaces to delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

Q*dir* Use the named *dir* as the queue directory.

r*time* Timeout reads after *time* interval.

S*file* Log statistics in the named *file*.

s Always instantiate the queue file, even if you are going to attempt immediate delivery. **sendmail** always instantiates the queue file before returning control the the client under any circumstances.

T*time* Set the queue timeout to *time*. After this interval, messages that have not been successfully sent will be returned to the sender.

t*S,D* Set the local timezone name to *S* for standard time and *D* for daylight time; this is only used under version six.

u*n* Set the default userid for mailers to *n*. Mailers without the *S* flag in the mailer definition will run as this user.

v	Run in verbose mode.

Mailer Description Flags

There are a number of options that can be specified as primitive flags (provided for compatibility with **delivermail**). These are the **e**, **i**, **m**, and **v** options. Also, the **f** option can be specified as the **-s** flag.

The following flags can be set in the mailer description.

f	The mailer wants a **-f** *from* flag, but only if this is a network forward operation (that is, the mailer will give an error if the executing user does not have special permissions).
r	Same as **f**, but sends a **-r** flag.
S	Don't reset the user ID before calling the mailer. This would be used in a secure environment where **sendmail** ran as root. This could be used to avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (for example, a user's **mail.cf** file).
n	Do not insert a UNIX system "From" line on the front of the message.
l	This mailer is local (that is, final delivery will be performed).
s	Strip quote characters off of the address before calling the mailer.
m	This mailer can send to multiple users on the same host in one transaction. When a **$u** macro occurs in the *argv* part of the mailer definition, that field will be repeated as necessary for all qualifying users.
F	This mailer wants a "From:" header line.
D	This mailer wants a "Date:" header line.
M	This mailer wants a "Message-id:" header line.
x	This mailer wants a "Full-Name:" header line.
P	This mailer wants a "Return-Path:" line.
u	Uppercase should be preserved in user names for this mailer.
h	Uppercase should be preserved in host names for this mailer.
A	This is an ARPANET-compatible mailer, and all appropriate modes should be set.
U	This mailer wants UNIX system "from" lines with the **uucp**-style "remote from <host>" on the end.
e	This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

X This mailer want to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This ensures that lines in the message containing a dot will not terminate the message prematurely.

L Limit the line lengths as specified in RFC821.

P Use the return-path in the SMTP "MAIL FROM:" command rather than just the return address.

I This mailer will be speaking SMTP to another **sendmail** — as such, it can use special protocol features. This option is not required (that is, if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).

C If mail is *received* from a mailer with this flag set, any addresses in the header that do not have an at sign ("@") after being rewritten by rule set three will have the "@domain" clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be automatically rewritten as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

For your convenience, refer to online **sendmail** manual pages (1M) for detailed descriptions of the options and flags that Apollo provides.

8.6 Normal Operation

8.6.1 The System Log

The system log is supported by the **syslog**(8) program.

Each line in the system log consists of a time stamp, the name of the machine that generated it (for logging from several machines over the ETHERNET), the word "sendmail:", and a message.

Levels

If you have **syslog**(8) or an equivalent installed, you will be able to do logging. There is a large amount of information that can be logged. The log is arranged as a succession of levels. At the lowest level, only extremely strange situations are logged. At the highest level, even the most mundane events are recorded. As a convention, log levels under 10 are considered "useful;" log levels above 10 are usually for debugging purposes.

8.6.2 The Mail Queue

The mail queue should be processed transparently. However, you can find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue can become clogged. Although **sendmail** ought to recover gracefully when the host comes up, you can find performance unacceptably bad in the meantime.

Printing the Queue

The contents of the queue can be printed by using the **mailq** command (or by specifying the **-bp** flag to **sendmail**):

The mailq Command

This will produce a listing of the queue IDs, the size of the message, the date the message entered the queue, and the sender and recipients.

Format of Queue Files

All queue files have the form *xfAA99999* where *AA99999* is the ID for this file and the *x* is a type. The types are

d The data file. The message body (excluding the header) is kept in this file.

l The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous **lf** file can cause a job to apparently disappear.

n This file is created when an ID is being created. It is a separate file to ensure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.

q The queue control file. This file contains the information necessary to process the job.

t A temporary file. These are an image of the **qf** file when it is being rebuilt. It should be renamed to a **qf** file very quickly.

x A transcript file, existing during the life of a session showing everything that happens during that session.

The **qf** file is structured as a series of lines each beginning with a code letter. The lines are as follows:

D The name of the data file. There can only be one of these lines.

H A header definition. There can be any number of these lines. The order is important; they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

R A recipient address. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient.

S The sender address. There can only be one of these lines.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M A message. This line is printed by the **mailq** command, and is generally used to store status information. It can contain any text.

As an example, the following is a queue file sent to "mckusick@calder" and "wnj":

```
DdfA13557
Seric
T404261372
P132
Rmckusick@calder
Rwnj
H?D?date: 23-Oct-82 15:49:32-PDT (Sat)
H?F?from: eric (Eric Allman)
H?x?full-name: Eric Allman
Hsubject: this is an example message
Hmessage-id: <8209232249.13557@UCBARPA.BERKELEY.ARPA>
Hreceived: by UCBARPA.BERKELEY.ARPA (3.227 [10/22/82]) id A13557;
23-Oct-82                15:49:32-PDT (Sat)
Hphone: (415) 548-3211
HTo: mckusick@calder, wnj
```

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

Forcing the Queue

In some cases, you may find that a major host going down for a couple of days can create a prohibitively large queue. This will result in **sendmail** spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
% cd /usr/spool
% mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
% /usr/lib/sendmail -oQ/usr/spool/omqueue -q
```

The **-oQ** flag specifies an alternate queue directory and the **-q** flag says to just run every job in the queue. Refer to Section 8.5 for details on configuration options.

When the queue is finally emptied, you can remove the directory:

```
% rmdir /usr/spool/omqueue
```

8.6.3 The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file **/usr/lib/aliases**. The aliases are of the form

name: name1, name2, ...

Only local names can be aliased. Aliases can be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a pound sign (#) are comments.

The second form is processed by the **dbm**(3) library. This form is in the files **/usr/lib/aliases.dir** and **/usr/lib/aliases.pag**. This is the form that **sendmail** actually uses to resolve aliases.

Rebuilding the Alias Database

The database can be rebuilt explicitly by executing the command

```
% newaliases
```

This is equivalent to giving **sendmail** the **-bi** flag:

```
% /usr/lib/sendmail -bi
```

If the **D** option (see Section 8.5 for details on configuration options) is specified in the configuration, **sendmail** will rebuild the alias database automatically if possible when it is out of date. The conditions under which it will do this are either of the following:

- The DBM version of the database is mode 666.
- **sendmail** is running setuid to root.

Auto–rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

Potential Problems

There are a number of problems that can occur with the alias database. They all result from a **sendmail** process accessing the database while it is only partially built. This can

happen under two circumstances: one process accesses the database while another process is rebuilding it, or the process rebuilding the database halts (due to being killed or a system crash) before completing the rebuild.

The **sendmail** facility has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form

@: @

(which is not normally legal). Before **sendmail** will access the database, it checks to ensure that this entry exists.

The **a** option is required in the configuration for this action to occur. This should normally be specified unless you are running **delivermail** in parallel with **sendmail.** It will wait up to five minutes for this entry to appear, at which point it will force a rebuild itself. Note that the **D** option must be specified in the configuration file for this operation to occur.

List Owners

If an error occurs on sending to a certain address, say "*x*", **sendmail** will look for an alias of the form "owner–*x*" to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
     sam@matisse
owner-unix-wizards: eric@ucbarpa
```

would cause "eric@ucbarpa" to get the error that will occur when someone sends to "unix-wizards" due to the inclusion of "nosuchuser" on the list.

8.6.4 Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user can put a file with the name **.forward** in his or her home directory. If this file exists, **sendmail** redirects mail for that user to the list of addresses listed in the **.forward** file. For example, if the home directory for user "mckusick" has a **.forward** file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for "mckusick" will be redirected to the specified accounts.

8.6.5 Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into **sendmail** that cannot be changed without changing the code. These built–in interpretations are described here.

Return–Receipt–To:

If this header is sent, a message will be sent to any specified addresses when the final delivery is complete, if the mailer has the **l** flag (local delivery) set in the mailer descriptor.

Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

Apparently-To:

If a message comes in with no recipients listed in the message (in a To:, Cc:, or Bcc: line) then **sendmail** will add an "Apparently-To:" header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC 822.

8.7 Tuning

There are a number of configuration parameters you can want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line "OT3d" sets option **T** to the value "3d" (three days).

8.7.1 Timeouts

All time intervals are set using a scaled syntax. For example, "10m" represents 10 minutes, whereas "2h30m" represents two and a half hours. The full set of scales is

s	seconds
m	minutes
h	hours
d	days
w	weeks

Queue Interval

The argument to the **-q** flag specifies how often a subdaemon will run the queue. This is typically set to between five minutes and one half hour.

Read Timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of large numbers of idle daemons piling up on your system. This timeout is set by using the **r** option in the configuration file.

Message Timeouts

After being in the queue for a few days, a message will time out. This is to ensure that at least the sender is aware of the inability to send a message. The timeout is typically set to three days. This timeout is set by using the **T** option in the configuration file.

The time of submission is set in the queue, rather than the amount of time left until timeout. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example,

```
% /usr/lib/sendmail -oT1d -q
```

will run the queue and flush anything that is one day old.

8.7.2 Delivery Mode

There are a number of delivery modes that **sendmail** can operate in, set by the **d** configuration option. These modes specify how quickly mail will be delivered. Legal modes are

i	Deliver interactively (synchronously)
b	Deliver in background (asynchronously)
q	Queue only (don't deliver)

There are tradeoffs. Mode **i** passes the maximum amount of information to the sender, but is hardly ever necessary. Mode **q** puts the minimum load on your machine, but means that delivery can be delayed for up to the queue interval. Mode **b** is probably a good compromise. However, this mode can cause large numbers of processes if you have a mailer that takes a long time to deliver a message.

8.7.3 Log Level

The level of logging can be set for **sendmail**. The default using a standard configuration table is level 9. The levels are as follows:

0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being deferred (due to a host being down, etc.).
5	Normal message queueups.
6	Unusual but benign incidents, for example, trying to process a locked queue file.
9	Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.
12	Several messages that are basically only of interest when debugging.
16	Verbose information regarding the queue.

8.7.4 File Modes

There are a number of files that can have a number of modes. The modes depend on what functionality you want and the level of security you require.

When to Use suid

The **sendmail** facility can safely be made setuid to root. At the point where it is about to **exec**(2) a mailer, it checks to see if the user ID is 0; if so, it resets the user ID and group ID to a default (set by the **u** and **g** options). (This can be overridden by setting the **S** flag to the mailer for mailers that are trusted and must be called as root.) However, this will cause mail processing to be accounted (using **sa**(8)) to root rather than to the user sending the mail.

Temporary File Modes

The mode of all temporary files that **sendmail** creates is determined by the **F** option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it will not be necessary to run **sendmail** as root at all (even when running the queue).

Should an Alias Database Be Writable?

The database that **sendmail** actually used is represented by the two files **aliases.dir** and **aliases.pag** (both in **/usr/lib**). The mode on these files should match the mode on **/usr/lib/aliases**. If **aliases** is writable and the DBM files (**aliases.dir** and **aliases.pag**) are not, users will be unable to reflect their desired changes through to the actual database. However, if **aliases** is read-only and the DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.

If your DBM files are not writable by the world or you do not have auto-rebuild enabled (with the **D** option), then you must be careful to reconstruct the alias database each time you change the text version:

```
% newaliases
```

If this step is ignored or forgotten, any intended changes will also be ignored or forgotten.

8.8 The Configuration File

This section describes the configuration file in detail, including hints on how to write one of your own if you have to. An overview of the configuration file is given first, followed by details of the semantics.

8.8.1 Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a pound sign (#) are comments.

R and S — Rewriting Rules

The core of address parsing are the rewriting rules. These are an ordered production system. **sendmail** scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the address is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and can be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are

S*n*

Sets the current rule set being collected to *n*. If you begin a rule set more than once, it deletes the old definition.

R lhs rhs *comments*

The fields must be separated by at least one tab character; there can be embedded spaces in the fields. The **lhs** is a pattern that is applied to the input. If it matches, the input is rewritten to the **rhs**. The *comments* are ignored.

D — Define Macro

Macros are named with a single character. These can be selected from the entire ASCII set, but user-defined macros should be selected from the set of uppercase letters only. Lowercase letters and special symbols are used internally.

The syntax for macro definitions is

D *xval*

where *x* is the name of the macro and *val* is the value it should have. Macros can be interpolated in most places by using the escape sequence **$***x*.

C and F — Define Classes

Classes of words can be defined to match on the left-hand side of rewriting rules. For example a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes can be given names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

The syntax is

C *cword1 word2...*
F *cfile* [*format*]

The first form defines the class `C` to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent. The second form reads the elements of the class **C** from the named file; the format is a **scanf**(3) pattern that should produce a single string.

M — Define Mailer

Programs and interfaces to mailers are defined in this line. The format is

M*name*, { *field*= *value*}*

where *name* is the name of the mailer (used internally only) and the "field=name" pairs define attributes of the mailer. Fields are

P[ath] The pathname of the mailer.

F[lags] Special flags for this mailer.

S[ender] A rewriting set for sender addresses.

R[ecipient] A rewriting set for recipient addresses.

A[rgv] An argument vector to pass to this mailer.

E[ol] The end-of-line string for this mailer.

M[axsize] The maximum message length to this mailer.

Only the first character of the field name is checked.

H — Define Header

The format of the header lines that **sendmail** inserts into the message are defined by the **H** line. The syntax of this line is

H[?*mflags*?] *hname*:*htemplate*

Continuation lines are reflected directly into the outgoing message. The *htemplate* is macro expanded before insertion into the message. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is reflected to the output regardless of these flags.

Some headers have special semantics that will be described below.

O — Set Option

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is

O *ovalue*

This sets option *o* to be *value*. Depending on the option, *value* can be a string, an integer, a boolean (with legal values **t**, **T**, **f**, or **F**; the default is TRUE), or a time interval.

T — Define Trusted Users

Trusted users are those users who are permitted to override the sender address by using the **-f** flag. These typically are root, **uucp**, and network, but for some users it can be convenient to extend this list to include other users, perhaps to support a separate login for each host. The syntax of this line is

T*user1 user2...*

There can be more than one of these lines.

P — Precedence Definitions

Values for the "Precedence:" field can be defined using the **P** control line. The syntax of this field is

P*name=num*

When the *name* is found in a "Precedence:" field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than 0 have the special property that error messages will not be returned. The default precedence is 0. For example, our list of precedences is

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

8.8.2 Semantics

This section describes the semantics of the configuration file.

Special Macros and Conditionals

Macros are interpolated using the construct $*x*, where *x* is the name of the macro to be interpolated. In particular, lower case letters are reserved to have special semantics, used to pass information in or out of **sendmail**, and some special characters are reserved to provide conditionals, etc.

The following macros *must* be defined to transmit information into **sendmail**:

e	The SMTP entry message
j	The "official" domain name for this site
l	The format of the UNIX system *from* line
n	The name of the daemon (for error messages)
o	The set of "operators" in addresses
q	The default format of sender address

The **$e** macro is printed out when SMTP starts up. The first word must be the **$j** macro. The **$j** macro should be in RFC821 format. The **$l** and **$n** macros can be considered constants except under terribly unusual circumstances. The **$o** macro consists of a list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if **r** were in the **$o** macro, then the input **address** would be scanned as three tokens: **add**, **r**, and ess. Finally, the **$q** macro specifies how an address should appear in a message when it is defaulted. For example,

```
De$j sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g $d
Do.:%@!^=/
Dq$g$?x ($x)$.
Dj$H.$D
```

An acceptable alternative for the **$q** macro is **$?x$x $.<$g>**. These correspond to the following two formats:

```
eric@Berkeley (Eric Allman)
Eric Allman <eric@Berkeley>
```

Some macros are defined by **sendmail** for interpolation into argv's for mailers or for other contexts. These macros are

a	The origination date in ARPANET format.
b	The current date in ARPANET format.
c	The hop count.
d	The date in the UNIX system (ctime) format.
f	The sender (from) address.
g	The sender address relative to the recipient.
h	The recipient host.

i	The queue ID.
p	**sendmail**'s *pid*.
r	Protocol used.
s	Sender's host name.
t	A numeric representation of the current time.
u	The recipient user.
v	The version number of **sendmail**.
w	The host name of this site.
x	The full name of the sender.
y	The IDof the sender's tty.
z	The home directory of the recipient.

There are three types of dates that can be used. The **$a** and **$b** macros are in ARPANET format; **$a** is the time as extracted from the "Date:" line of the message (if there was one), and **$b** is the current date and time (used for postmarks). If no "Date:" line is found in the incoming message, **$a** is set to the current time also. The **$d** macro is equivalent to the **$a** macro in the UNIX system (*ctime*) format.

The **$f** macro is the ID of the sender as originally determined; when mailing to a specific host the **$g** macro is set to the address of the sender relative to the recipient. For example, if "bollard@matisse" is sent from the machine "ucbarpa" the **$f** macro will be "eric" and the **$g** macro will be "eric@ucbarpa."

The **$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to **sendmail**. The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the **/etc/passwd** file.

When sending, the **$h**, **$u**, and **$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the $@ and **$:** part of the rewriting rules, respectively.

The **$p** and **$t** macros are used to create unique strings (for example, for the "Message-id:" field). The **$i** macro is set to the queue ID on this host; if put into the time stamp line it can be extremely useful for tracking messages. The **$y** macro is set to the ID of the terminal of the sender (if known); some systems like to put this in the UNIX system "From" line. The **$v** macro is set to be the version number of **sendmail**; this is normally put in time stamps and has been proven extremely useful for debugging. The **$w** macro is set to the name of this host if it can be determined. The **$c** field is set to the "hop count," that is, the number of times this message has been processed. This can be determined by the **-h** flag on the command line or by counting the time stamps in the message.

The **$r** and **$s** fields are set to the protocol used to communicate with **sendmail** and the sending host name; these are not supported in the current version.

Conditionals can be specified by using the syntax:

$?x *text1* $| *text2* $.

This interpolates *text1* if the macro $x is set, and *text2* otherwise. The "else" ($|) clause can be omitted.

Special Classes

The class $=w is set to be the set of all names this host is known by. This can be used to delete local host names.

The Left-Hand Side

The left-hand side (LHS) of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced by using a dollar sign. The metasymbols are

$*	Match 0 or more tokens.
$+	Match one or more tokens.
$-	Match exactly one token.
$=*x*	Match any token in class *x*.
$~*x*	Match any token not in class *x*.

If any of these match, they are assigned to the symbol **$***n* for replacement on the right-hand side (RHS), where *n* is the index in the LHS. For example, if the LHS:

```
$-:$+
```

is applied to the input:

```
UCBARPA:eric
```

the rule will match, and the values passed to the RHS will be:

```
$1 UCBARPA
$2 eric
```

The Right-Hand Side

When the right-hand side of a rewriting rule matches, the input is deleted and replaced by the right-hand side. Tokens are copied directly from the RHS unless they are begin with a dollar sign. Metasymbols are as follows:

$*n*	Substitute indefinite token *n* from LHS.
$>*n*	"Call" rule set *n*.
$#*mailer*	Resolve to *mailer*.
$@*host*	Specify *host*.
$:*user*	Specify *user*.

The **$***n* syntax substitutes the corresponding value from a **$+**, **$-**, **$***, **$=**, or **$~** match on the LHS. It can be used anywhere.

The **$>** *n* syntax causes the remainder of the line to be substituted as usual and then passed as the argument to rule set *n*. The final value of rule set *n* then becomes the substitution for this rule.

The **$#** syntax should *only* be used in rule set 0. It causes evaluation of the rule set to terminate immediately, and signals to **sendmail** that the address has completely resolved. The complete syntax is

$#***mailer***$@*host***$:***user*

This specifies the {mailer, host, user} 3-tuple necessary to direct the mailer. If the mailer is local, the host part can be omitted. The *mailer* and *host* must be a single word, but the *user* can be multipart.

An RHS can also be preceeded by a **$@** or a **$:** to control evaluation. A **$@** prefix causes the rule set to return with the remainder of the RHS as the value. A **$:** prefix causes the rule to terminate immediately, but the rule set to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes can precede a **$>** spec; for example:

```
R$+   $:$>7$1
```

matches anything, passes that to rule set seven, and continues; the **$:** is necessary to avoid an infinite loop.

Rule Sets Applied to Recipient Addresses

As shown in Figure 8-1, there are five rewriting sets that have specific semantics.

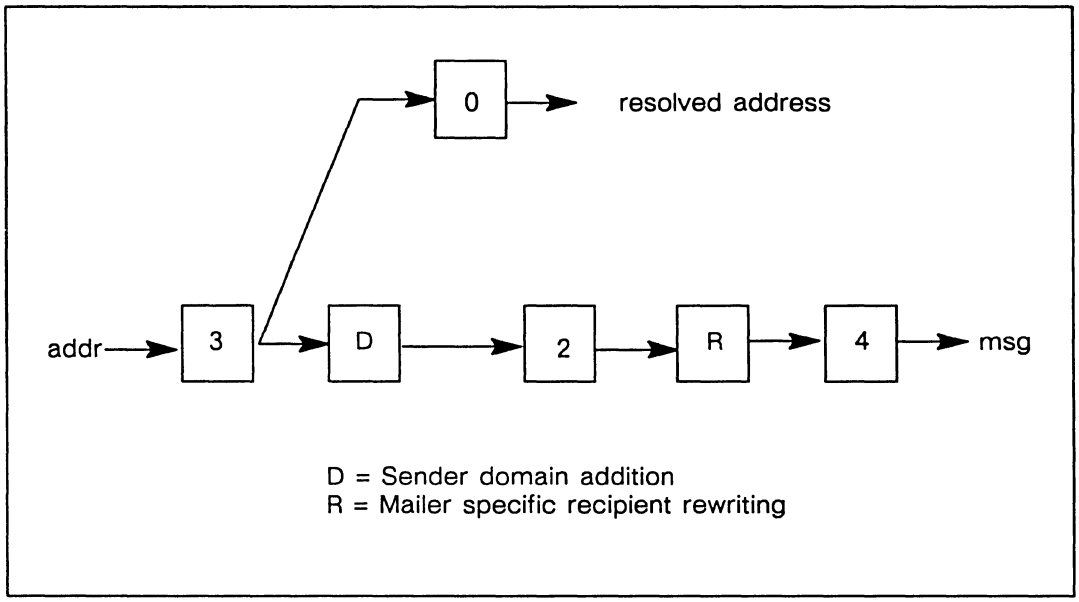


Figure 8–1. Rewriting Set Semantics for Recipient Addresses

The following rule sets are applied to recipient addresses:

- Rule set 3 turns an address into "canonical form," that is, a form that is ready for internal use. This form should have the basic syntax:

  local-part@host-domain-spec

  If no "@" sign is specified, then the host-domain-spec *can* be appended from the sender address (if the C flag is set in the mailer definition corresponding to the *sending* mailer). Rule set 3 is applied by **sendmail** before doing anything with any address.

- Rule set 0 is applied after rule set 3 to addresses that are going to actually specify recipients. It must resolve to a {*mailer, host, user*} triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the **$h** macro for use in the *argv* expansion of the specified mailer.

- Rule set 2 is globally applied to all recipient addresses in the mail header fields: *To:*, *Cc:*, *Apparently-to:*, and *Bcc:*. They are applied before any specification in the mailer definition. They must never resolve.

- Rule set 4 is applied to all addresses in the message. It is typically used to translate internal to external form by removing any "internal-only" additions prior to sending mail to the "outside" world.

Rule Sets Applied to Sender Addresses

As shown in Figure 8–2, there are five rewriting sets that have specific semantics.

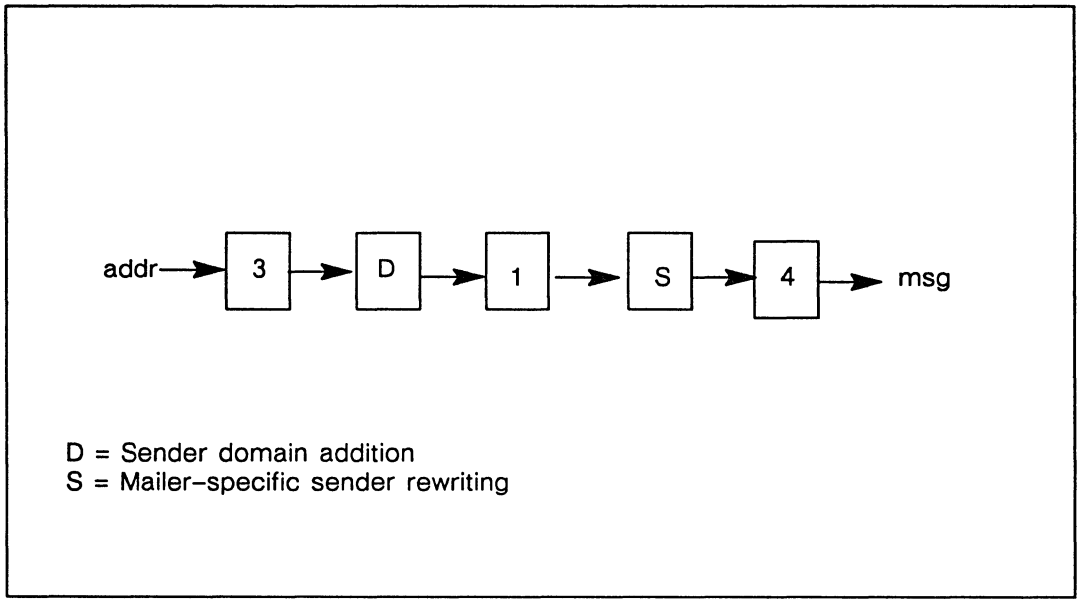


Figure 8-2. Rewriting Set Semantics for Sender Addresses

The following rule sets are applied to sender addresses:

- Rule set 3 turns an address into "canonical form," that is, a form that is ready for internal use. This form should have the basic syntax:

  local-part@host-domain-spec

  If no "@" sign is specified, then the host-domain-spec *can* be appended from the sender address (if the **C** flag is set in the mailer definition corresponding to the *sending* mailer). rule set three is applied by **sendmail** before doing anything with any address.

- Rule set 1 is globally applied to the sender address in either of the mail header field: *From:* or *Apparently-from:*. It is applied before any specification in the mailer definition and must never resolve.

- Rule set 4 is applied to all addresses in the message. It is typically used to translate internal to external form by removing any "internal-only" additions prior to sending mail to the "outside" world.

Mailer Flags

There are a number of flags that can be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. Refer to Section 8.6 for details.

The "error'' Mailer

The mailer with the special name "error" can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated if the LHS matches. This mailer is only functional in rule set 0.

8.8.3 Building a Configuration File from Scratch

Building a configuration table from scratch is an extremely difficult job. Fortunately, it is almost never necessary to do so; nearly every situation that can come up can be resolved by changing an existing table. In any case, it is critical that you understand what it is that you are trying to do and come up with a philosophy for the configuration table. This section is intended to explain what the real purpose of a configuration table is and to give you some ideas for what your philosophy might be.

What You Are Trying to Do

The configuration table has three major purposes. The first and simplest is to set up the environment for **sendmail**. The secondis to rewrite addresses in the message. This should be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in rule set 3. The second phase maps this canonical form into the syntax appropriate for the receiving mailer.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered.

Philosophy

The particular philosophy you choose will depend heavily on the size and structure of your organization.

One general point applies to all philosophies: it is almost always a mistake to try to do full name resolution. For example, if you are trying to get names of the form *user@host* to the ARPANET, it does not pay to route them to **xyzvax!decvax!ucbvax!c70:***user@host* since you then depend on several links not under your control. The best approach to this problem is to simply forward to **xyzvax!***user@host* and let **xyzvax** worry about it from there. In summary, just get the message closer to the destination, rather than determining the full path.

Large Site, Many Hosts — Minimum Information

You can have decided that the only reasonable philosophy in our environment is to designate one host as the guru for your site. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they require. In addition, any information they do have should be hints rather than solid information.

For example, a typical site on a local ETHERNET is "monet." The site **monet** has a list of known ETHERNET hosts; if it receives mail for any of them, it can do direct delivery. If it

receives mail for any unknown host, it just passes it directly to "ucbvax," the master host. The master host **ucbvax** can determine that the host name is illegal and reject the message, or can be able to do delivery. However, it is important to note that when a new ETHERNET host is added, the only host that *must* have its tables updated is **ucbvax**; the others *can* be updated as convenient, but this is not critical.

This picture is slightly muddied due to network connections that are not actually located on **ucbvax**. For example, the TCP connection is currently on "ucbarpa." However, **monet** *does not* know about this; the information is hidden totally between **ucbvax** and **ucbarpa**. Mail going from **monet** to a TCP host is transferred via the ETHERNET from **monet** to **ucbvax**, then via the ETHERNET from **ucbvax** to **ucbarpa**, and then is submitted to the ARPANET. This can be an acceptable tradeoff.

An interesting point is that it would be possible to update **monet** to send TCP mail directly to **ucbarpa** if the load got too high; if **monet** failed to note a host as a TCP host it would go via **ucbvax** as before, and if **monet** incorrectly sent a message to **ucbarpa** it would still be sent by **ucbarpa** to **ucbvax** as before. The only problem that can occur is loops, as if **ucbarpa** thought that **ucbvax** had the TCP connection and vice versa. For this reason, updates should *always* happen to the master host first.

This philosophy results as much from the need to have a single source for the configuration files (typically built using **m4**(1) or some similar tool) as any logical need. Maintaining more than three separate tables by hand is essentially an impossible job.

Small Site — Complete Information

A small site (two or three hosts) can find it more reasonable to have complete information at each host. This would require that each host know exactly where each network connection is, possibly including the names of each host on that network. As long as the site remains small and the the configuration remains relatively static, the update problem will probably not be too great.

Single Host

This is in some sense the trivial case. The only major issue is trying to ensure that you don't have to know too much about your environment. For example, if you have a **uucp** connection you might find it useful to know about the names of hosts connected directly to you, but this is really not necessary since this can be determined from the syntax.

Relevant Issues

The following characters have special interpretations:

```
< > ( ) " \
```

Essentially, each host is given a name which is a right-to-left dot qualified pseudopath from a distinguished root. The elements of the path need not be physical hosts; the domain is logical rather than physical. For example, at Berkeley one legal host is "a.cc.berkeley.arpa"; reading from right to left, "arpa" is a top-level domain (related to, but not limited to, the physical ARPANET), "berkeley" is both an ARPANET host and a logical domain which is actually interpreted by a host called **ucbvax** (which is actually just the "major" host for this domain), "cc" represents the Computer Center, (in this case a

strictly logical entity), and "a" is a host in the Computer Center; this particular host happens to be connected via berknet, but other hosts might be connected via one of two ETHERNETs or some other network.

How to Proceed

Once you have decided on a philosophy, it is worth examining the available configuration tables to decide if any of them are close enough to steal major parts. Even under the worst of conditions, there is a fair amount of boiler plate that can be collected safely.

The next step is to build rule set 3. This will be the hardest part of the job. Beware of doing too much to the address in this rule set, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with addresses with no domain spec at all. Since **sendmail** likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also try to avoid fully qualifying domains in this rule set. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages.

Once you have rule set 3 finished, the other rule sets should be relatively trivial. If you need hints, examine the supplied configuration tables.

Testing The Rewriting Rules: The -bt Flag

When you build a configuration table, you can do a certain amount of testing using the "test mode" of **sendmail**. For example, you could invoke **sendmail** as

```
% sendmail -bt -Ctest.cf
```

which would read the configuration file **test.cf** and enter test mode. In this mode, you enter lines of the form:

rwset address

where *rwset* is the rewriting set you want to use and *address* is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You can use a comma separated list of *rwset*s for sequential application of rules to an input; rule set 3 is always applied first. For example:

1,21,4 monet:bollard

first applies rule set 3 to the input **monet:bollard**. rule set 1 is then applied to the output of rule set 3, followed similarly by rule sets 21 and 4.

If you need more detail, you can also use the **-d21** flag to turn on more debugging. For example,

```
% sendmail -bt -d21.99
```

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

Building Mailer Descriptions

To add an outgoing mailer to your mail system, you will have to define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names **local** and **prog** must be defined.

The pathname of the mailer must be given in the **P** field. If this mailer should be accessed via an IPC connection, use the string **[IPC]** instead.

The **F** field defines the mailer flags. You should specify an **f** or **r** flag to pass the name of the sender as a **-f** or **-r** flag respectively. These flags are only passed if they were passed to **sendmail**, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky you can just specify **-f $g** in the *argv* template. If the mailer must be called as **root** the **S** flag should be given; this will not reset the user ID before calling the mailer. **sendmail** must be running setuid to root for this to work.

If this mailer is local (that is, will perform final delivery rather than another network hop) the **l** flag should be given. Quote characters (backslashes and " characters) can be stripped from addresses if the **s** flag is specified; if this is not given they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction the **m** flag should be stated. If this flag is on, then the *argv* template containing **$u** will be repeated for each unique user on a given host. The **e** flag will mark the mailer as being "expensive," which will cause **sendmail** to defer connection until a queue run. The **c** configuration option must be given for this to be effective.

An unusual case is the **C** flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain spec of the sender (that is, the "@host.domain" part) is saved and is appended to any addresses in the message that do not already contain a domain spec. For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

will be modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

if and only if the **C** flag is defined in the mailer corresponding to **eric@ucbarpa**.

The **S** and **R** fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (rule set 4) is applied. A typical use is to append the current domain to addresses that do not already have a domain. For example, a header of the form:

```
From: eric
```

might be changed to be:

```
From: eric@ucbarpa
```

or

```
From: ucbvax!eric
```

depending on the domain it is being shipped into. These sets can also be used to do special–purpose output rewriting in cooperation with rule set 4.

The **E** field defines the string to use as an end–of–line indication. A string containing only newline is the default. The usual backslash escapes (**\r**, **\n**, **\f**, **\b**) can be used.

Finally, an *argv* template is given as the **E** field. It can have embedded spaces. If there is no *argv* with a **$u** macro in it, **sendmail** will speak SMTP to the mailer. If the pathname for this mailer is **[IPC]**, the *argv* should be

IPC $h [*port*]

where *port* is the optional port number to connect to.

For example, the specifications:

```
Mlocal, P=/bin/mail, F=rlsm S=10, R=20, A=mail -d $u
Mether, P=[IPC],   F=meC, S=11, R=21, A=IPC $h, M=100000
```

specifies a mailer to do local delivery and a mailer for ETHERNET delivery. The first is called **local**, is located in the file **/bin/mail**, takes a picky **-r** flag, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; rule set 10 should be applied to sender addresses in the message and rule set 20 should be applied to recipient addresses; the *argv* to send to a message will be the word **mail**, the word **-d**, and words containing the name of the receiving user. If a **-r** flag is inserted it will be be connected to via an IPC connection, it can handle multiple users at once, connections should be deferred, and any domain from the sender address should be appended to any receiver name without a domain; sender addresses should be processed by rule set 11 and recipient addresses by rule set 21. There is a 100,000 byte limit on messages passed through this mailer.

8.9 Summary of Support Files

This is a summary of the support files that **sendmail** creates or generates.

/usr/lib/sendmail	The binary of **sendmail**.
/usr/bin/newaliases	A link to **/usr/lib/sendmail**; causes the alias database to be rebuilt. Running this program is completely equivalent to giving **sendmail** the **-bi** flag.
/usr/bin/mailq	Prints a listing of the mail queue. This program is equivalent to using the **-bp** flag to **sendmail**.
/usr/lib/sendmail.cf	The configuration file, in textual form.
/usr/lib/sendmail.fc	The configuration file represented as a memory image.
/usr/lib/sendmail.hf	The SMTP help file.
/usr/lib/sendmail.st	A statistics file; need not be present.

/usr/lib/aliases	The textual version of the alias file.
/usr/lib/aliases.{pag,dir}	The alias file in **dbm**(3) format.
/etc/syslog	The program to do logging.
/etc/syslog.conf	The configuration file for **syslog**.
/etc/syslog.pid	Contains the process id of the currently running **syslog**.
/usr/spool/mqueue	The directory in which the mail queue and temporary files reside.
/usr/spool/mqueue/qf*	Control (queue) files for messages.
/usr/spool/mqueue/df*	Data files.
/usr/spool/mqueue/lf*	Lock files.
/usr/spool/mqueue/tf*	Temporary versions of the **qf** files, used during queue file rebuild.
/usr/spool/mqueue/nf*	A file used when creating a unique ID.
/usr/spool/mqueue/xf*	A transcript of the current session.

Chapter 9

Administrative Commands

Contents

NAME

intro – introduction to system administration commands

DESCRIPTION

This chapter contains commands and programs that perform miscellaneous system maintenance and administration functions. These commands are labeled with the number 8, both to distinguish the commands from those in the *BSD Command Reference*, and to maintain some continuity with the numbering scheme of the original *UNIX Programmers' Manuals*. However, the pages are numbered in the form 9-*n*, where *n* is the command's page number in this chapter (9).

You must be logged in as the super-user to use many of these utilities.

The examples in this section use the Bourne shell default prompt ($).

NAME

Uutry – try to contact remote system with debugging on

SYNOPSIS

/usr/lib/uucp/Uutry [**–x** *debug_level*] [**–r**] *system_name*

DESCRIPTION

Uutry is a shell that is used to invoke **uucico** to call a remote site. Debugging is turned on (default is level 5);

OPTIONS

–x*debug_level*	Overrides the default debugging level (level 5.) The level is set to *debug_level*.
–r	Overrides the retry time in **/usr/spool/uucp/.status**. The debugging output is put in file **/tmp/***system_name*. A tail **–f** of the output is executed. DELETE or BREAK gives control back to the terminal while **uucico** continues to run, putting its output in **/tmp/***system_name*.

FILES

/usr/lib/uucp/Systems

/usr/lib/uucp/Permissions

/usr/lib/uucp/Devices

/usr/lib/uucp/Maxuuxqts

/usr/lib/uucp/Maxuuscheds

/usr/spool/uucp/*

/usr/spool/locks/LCK*

/usr/spool/uucppublic/*

/tmp/system_name

SEE ALSO

uucico(1M).
uucp(1C), uux(1C)

NAME

ac – login accounting

SYNOPSIS

/etc/ac [–w *wtmp* **] [–p] [–d] [** *people* **] ...**

DESCRIPTION

ac produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced. If you don't specify a *wtmp* file, **ac** uses **/usr/adm/wtmp**.

The accounting file **/usr/adm/wtmp** is maintained by **init** and **login**. Neither of these programs creates the file, so if the file does not exist no connect-time accounting is done. To start accounting, create the *wtmp* file with length 0. If the file is left undisturbed it will grow without bound, so you should periodically collect any information desired and truncate the file.

OPTIONS

–w *wtmp*	Specify an alternate *wtmp* file.
–p	Print individual totals; without this option, only totals are printed.
–d	Causes a printout for each midnight to midnight period.
people	Limit the printout to only the specified login names.

FILES

/usr/adm/wtmp

SEE ALSO

init(8), sa(8), login(1), utmp(5)

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*
arp **–a**
arp **–d** *hostname*
arp **–s** *hostname ether_addr* [**temp**] [**pub**] [**trail**]
arp **–f** *filename*

DESCRIPTION

The **arp** program displays and modifies the Internet-to-ETHERNET* address translation tables used by the address resolution protocol (**arp**(4p)).

With no flags, the program displays the current ARP entry for *hostname*. You may specify the host by name or by number, using Internet dot notation.

OPTIONS

–a Display all of the current ARP entries in the internal file.

–d *hostname* A super-user may delete an entry for the host called *hostname*.

–s *hostname ether_addr* [**temp**] [**pub**] [**trail**]
Create an ARP entry for the host called *hostname* with the ETHERNET address *ether_addr*. The ETHERNET address is given as six hex bytes separated by colons. The entry will be permanent unless you specify the word **temp** in the command. If you specify the word **pub**, the entry will be ''published''; that is, this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

–f *filename* Read the file *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form

hostname ether_addr [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

NOTES

ETHERNET is a registered trademark of Xerox Corporation.

SEE ALSO

inet(3N), arp(4P), ifconfig(8C);
Configuring and Managing TCP/IP.

NAME

chown – change owner

SYNOPSIS

/etc/chown [**–f –R**] *owner*[*.group*] *file* ...

DESCRIPTION

The **chown** command changes the owner of the *files* to *owner*. The owner may be expressed either as a decimal UID (user ID) or as a login name found in the password file. An optional group may also be specified. The group may be expressed either as a decimal GID (group ID) or as a group name found in the group ID file.

Only the super-user can change owner, in order to simplify accounting procedures.

OPTIONS

–f Don't report errors (force).

–R Recursively descend the directory arguments, setting the specified owner. When symbolic links are encountered they are not traversed. On some systems, the **–R** option changes the ownership of symbolic links. Since BSD doesn't support symbolic link ownership, **chown –R** doesn't affect symbolic links.

FILES

/etc/passwd

SEE ALSO

chgrp(1), chown(2), passwd(5), group(5)

NAME

comsat – biff server

SYNOPSIS

/etc/comsat

DESCRIPTION

comsat is the server process which receives reports of incoming mail and notifies users if they have requested this service. **comsat** receives messages on a datagram port associated with the ''biff'' service specification (see **services**(5) and **inetd**(8)). The one line messages are of the form

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a **biff y**), the *offset* is used as a seek offset into the appropriate mailbox file and the first seven lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header, other than the ''From'', ''To'', ''Date'', or ''Subject'' lines, are not included in the displayed message.

FILES

/etc/utmp to find out who's logged on and on what terminals

CAUTIONS

The message header filtering is prone to error. The density of the information presented is near the theoretical minimum.

Users should be notified of mail which arrives on machines other than the one to which they are currently logged in.

The notification should appear in a separate window.

SEE ALSO

biff(1), inetd(8)

NAME

cpboot – copy the system boot file sysboot

SYNOPSIS

/**etc/cpboot** *source_directory target_directory*

DESCRIPTION

cpboot copies the system boot file **sysboot** from one directory to another. The **sysboot** file is used by the bootstrap prom to start the system. **cpboot** is useful for copying **sysboot** to a floppy disk, thus making the standalone utilities (**sau**) directory on the floppy disk accessible from the boot prom. You may also use it to update a Winchester disk when a new software release is distributed.

source_directory (required) Specify directory containing the file **sysboot**.

target_directory (required) Specify directory to which **sysboot** is to be copied. This must be the entry directory on the target logical volume.

NAME

cron – clock daemon

SYNOPSIS

/etc/cron

DESCRIPTION

The **cron** command executes commands at the dates and times specified in the files **/usr/lib/crontab**, and **/usr/lib/crontab.local**. None, either one, or both of these files may be present. Since **cron** doesn't exit, it should be executed only once. This is best done by running **cron** from the initialization process, using an entry in the **/etc/rc** file.

The **crontab** files consist of lines with seven fields each; the fields are separated by spaces or tabs. The first five fields contain integer patterns that specify the time and date in the following format:

Minute (0-59)
Hour (0-23)
Day of the month (1-31)
Month of the year (1-12)
Day of the week (1-7, 1 is Monday)

Each of these patterns may contain

- A number in the range above
- Two numbers separated by a minus sign, indicating an inclusive range
- A list of numbers separated by commas, meaning any of the numbers
- An asterisk, meaning all legal values for that field.

For example, if the month field contains:

1,3,5

cron will execute at all times in January, March, and May when the other fields' values are true.

The sixth field is a username; **cron** will execute the command with the permissions and uid of the specified user.

The seventh field consists of all the text on a line following the sixth field, including spaces and tabs. This text is treated as a command that is executed by the Shell at the time(s) and date(s) specified. A percent character (%) in this field is translated to a new-line character.

cron checks the files **/usr/lib/crontab** and **/usr/lib/crontab.local** every minute, on the minute.

EXAMPLE

A line of the form

```
0,15,30,45 * * * * root /usr/lib/atrun
```

in the **/usr/lib/crontab** file will run the **atrun** program every 15 minutes.

FILES

/usr/lib/crontab	
/usr/lib/crontab.local	files of times, dates, and commands to be run

NAME

crpty – create pseudo tty device entries

SYNOPSIS

/etc/crpty *count*

DESCRIPTION

crpty creates the pairs of device entries that are used by the pseudo terminal driver; see **pty**(4). **crpty** creates device pairs **/dev/ptyp[0-f]** and **/dev/ttyp[0-f]**. **crpty** takes the argument *count*, which is the number of pairs of **pty** entries to create. The maximum number of pairs allowed is 16.

The number of psuedo-terminal device pairs created controls the number of simultaneous incoming **rlogin** and **telnet** processes to an individual node.

EXAMPLE

crpty 2

creates **/dev/ptyp0**, **/dev/ttyp0**, and **/dev/ptyp1**, **/dev/ttyp1**.

SEE ALSO

pty(4)

NAME

ctnode – catalog a node in the network

SYNOPSIS

/etc/ctnode [*options*] [*node_name* [*net.*]*node_id* ...]

DESCRIPTION

ctnode informs the local node that a remote node exists, thereby enabling network file access to the remote node. The command catalogs the *node_name* in the local copy of the network root directory as the entry directory for the remote node. In other words, **ctnode** adds the directory *//node_name* to your copy of the network root directory.

We assign a node ID to every node during the manufacturing process. To find out the node ID of a node, type the following command at its keyboard:

```
$ lcnode -me
```

from another node's network root into your own, or any other node's network root. The merge options (**–md** and **–ms**) add the entry for a node to the target, provided the entry does not already exist and the source has exactly one entry for that node. In the case of one source and one target entry that match for a node, those entries are assumed to be correct. All other cases are considered to be ambiguous and the "confusion-resolution protocol" is invoked.

This "confusion-resolution protocol" first attempts to verify the correct entry name with the node itself. If the node is available, the reply from the node is cataloged regardless of whether **–md** or **–ms** is used because an answer from the node itself is assumed to be the truth.

If the node is unavailable to resolve an ambiguity, the entry containing the most recent UID (latest time stamp portion of the UID) is used. In this case, existing entries in the target directory are only updated if the **–ms** option is used. Multiple name/ID pairs are permitted.

If you do not specify **–n**, **–update**, or **–from**, the *node_name* and *net.node_id* arguments are required.

node_name (optional) Specify the name of the node you wish to catalog. If the *net.node_id* argument is specified, then *node_name* is required.

Default if omitted: you must use **–n, –update**, or **–from**

[*net.*]*node_id* (optional)

Specify the hexadecimal ID (and optional network ID) of the node you wish to catalog. The node must be connected to the network when this command is executed. If the *node_name* argument is specified, then *node_id* or [*net*].*node_id* is required.

Default if omitted: you must use **–n, –update**, or **–from**

Multiple name/ID pairs are permitted.

OPTIONS

If you do not specify **–n, –update**, or **–from**, the *node_name* and [*net.*]*node_id* arguments are required. The **–n, –update**, and **merge** options work only for remote nodes running Aegis SR5.0 or later. The [*net.*]*node_id* forms work only when both the local and remote nodes run Aegis SR9.0 or later.

–root Catalog *node_name* as the entry directory name for *node_id* in both the master network root directory and the local copy of the network root directory. This option is valid only if the *node_name* and *node_id* arguments are specified. This option is not valid if the **–n** option is specified.

–n [*net.*]*node_id...* Copy the entry directory name from the network root directory of the specified remote node to the network root directory of the local node. You do not need to know the entry directory name. However, you must specify the *node_id* or the *net.node_id* of the remote node. Multiple *node_id*'s and *net.node_id*'s may be specified. Use this option instead of the *node_name* net.*node_id* argument pair. This option is not valid if the **–r** option is specified.

–update Obtain a list of nodes currently responding to a network inquiry and perform the same operation as **–n** for each node. Names are replaced with the most current version, if they already exist in your local copy of the network root directory, and new names are added.

–from *//node ...* Look in the specified list of network root directories for the names to add to the target network root, or use this network root as the source for names to merge into the target network root. Wildcards may be used to specify source node names. The **–from** option is not supported in a Domain internet environment.

–md This option is used with **–from**. Merges all names in the source network root into the target network root. Preference is given to existing names in the target if there are unresolved conflicts (see the discussion of "confusion-resolution protocol" above).

–ms This option is the same as **–md**, except that preference is given to entries in the source network root when there are unresolved conflicts (see the discussion of "confusion-resolution protocol" above).

–on *//node ...* Catalog names in the network root of the specified nodes instead of the local network root. Wildcards may be used to specify target node names. The **–on** option is not supported in a Domain internet environment.

–r Replace cataloged names if they already exist. An error occurs if you do not specify this option and try to add a *node_name* that has already been cataloged (unless you are using **–update**).

–l List node names as they are cataloged.

–idupl Ignore entry (suppress error messages) if name already exists in the target.

–lc List invocations and resolutions of the "confusion-resolution protocol".

EXAMPLES

Add the node whose ID is 21 and whose entry directory name is **os** to your node's catalog:

$ **/etc/ctnode os 21**

Bring your node's catalog up to date with any new nodes on the network:

$ **/etc/ctnode –update**

Copy names **os** and **eve** from the network root on **//master**.

$ **/etc/ctnode os eve –from //master**

Add node ID 21 with the name os to the network root of all nodes whose names begin with "a".

$ **/etc/ctnode os 21 –on //a?***

Merge network root of os into local network root, resolving conflicts:

$ **/etc/ctnode –md –from //os**

NAME

drm_admin – Data Replication Manager Administrative Tool

SYNOPSIS

/etc/ncs/drm_admin

DESCRIPTION

The **drm_admin** tool administers servers based on the Data Replication Manager (DRM), such as the Global Location Broker (GLB).

It can inspect or modify replica lists, merge databases to force convergence among replicas, stop servers, and delete replicas.

The role of **drm_admin** is to administer the replication of databases, not to change the data they contain. For instance, you can use **drm_admin** to merge two replicas of the GLB database, but you must use **lb_admin** to add a new entry to the database. Also, although **drm_admin** can stop or delete a GLB replica, you must invoke **glbd** (the GLB daemon) directly if you want to start or create a replica.

Once invoked, **drm_admin** enters an interactive mode, in which it accepts the commands described in the following section.

COMMANDS

Most **drm_admin** commands operate on a default object (*default_obj*) at a default host (*default_host*). Together, *default_obj* and *default_host* specify a default replica. Defaults are established by the **set** command and are remembered until changed by another **set**.

Currently, the only known object is **glb**.

Some **drm_admin** commands operate on a host other than the default; we identify this host as *other_host*. The host name you supply as a *default_host* or an *other_host* takes the form *family*:*host*. The only currently supported *family* is **dds**; you can specify a host in this family by its entry directory or by its network address. For example, **dds://thurber** and **dds:#1234.abcd** are acceptable host names.

addrep *other_host* Add *other_host* to the replica list at *default_host*. The replica at *default_host* will propagate *other_host* to all other replica lists for *default_obj*.

delrep *other_host* [**–force**]
Delete the replica of *default_obj* at *other_host*.

The **delrep** command tells the replica at *other_host*

1. To propagate all of the entries in its propagation queue
2. To propagate a delete request to all other replicas, causing *other_host* to be deleted from all other replica lists for *default_obj*

3. To delete its copy of *default_obj*
4. To stop running

The **–force** option causes a more drastic delete. It deletes *other_host* from the replica list at *default_host*. The replica at *default_host* propagates the delete request to the replicas at the hosts remaining on its list, thereby removing *other_host* from all other replica lists for *default_obj*.

A force delete can cause data to be lost and should only be used when a replica has irrevocably "died." We recommend strongly that you do a **merge_all** operation after the force delete to prevent the remaining replicas of the *default_obj* database from becoming inconsistent. If the deleted replica is still running, it should be **reset**.

info Get status information about the replica for *default_obj* at *default_host*.

lrep [–d] [–clocks] [–na]
List replicas for *default_obj* as stored in the replica list at *default_host*.

The **–d** option lists deleted as well as existing replicas.

The **–clocks** option shows the current time on each host and indicates clock skew among the replicas.

The **–na** option lists the network address of each host.

merge { –from | –to } *other_host*
The **merge** command copies entries in the *default_obj* database and replica list from one replica to another. It copies an entry if no corresponding entry exists in the destination database or if the corresponding entry in the destination database bears an earlier time stamp.

A merge does not cause entries to be propagated. The database and replica list at the origination are not changed.

The **–from** option copies entries from the *default_obj* database and replica list at *other_host* to the *default_obj* database and replica list at *default_host*.

The **–to** option copies entries from the database and replica list at *default_host* to the database and replica list at *other_host*.

A **merge –from** followed by a **merge –to** causes the replicas at the two hosts to converge.

merge_all The **merge_all** command uses *default_host* as the hub for a global merge of all replicas for *default_obj*. A **merge_all** first does a **merge –from** each host on *default_host*'s replica list; then it does a **merge –to** each host on the replica list. All replicas of *default_obj* are thereby forced into a consistent state.

A **merge_all** should be used

- When a replica is force deleted
- When a replica is reset
- When a replica has been incommunicado for 2 weeks or more
- When a replica "dies" (for example, when its database is destroyed by a disk failure). The **merge_all** operation does not cause any entries to be propagated.

monitor [**–r** *n*] This command causes **drm_admin** to read the clock of each replica of the *default_obj* every *n* minutes and to report any clock skews or non-answering replicas. If you do not specify **–r**, The period is 15 minutes.

quit Quit the **drm_admin** session.

reprep *other_host* Replace the network address for *other_host* in the replica list at *default_host*. The replica at *default_host* will propagate the new entry for *other_host* to all other replica lists for *default_obj*. Use **reprep** only when a host's network number changes.

reset *other_host* Reset the replica of *default_obj* at *other_host*.

The **reset** command tells the replica at *other_host* to delete its copy of *default_obj* and to stop running. It does not cause *other_host* to be deleted from any other replica lists. This command can cause data to be lost unless a successful **merge_all** is done first.

set [**–o** *obj_name*] **–h** *host_name*
Set the default object and host. Subsequent commands that do not specify a host will be sent to this host. All subsequent commands will operate on the object *obj_name*. If you do not specify the **–o** option, **drm_admin** keeps the current *default_obj*.

If you use **set** with the **–o** option, **drm_admin** checks the clocks at all hosts with replicas of the specified object.

stop Stop the server for *default_obj* that is running at *default_host*.

EXAMPLES

Start **drm_admin**, set the default object to **glb**, and set the default host to //**mars**:

```
$ /etc/ncs/drm_admin
drm_admin: set -o glb -h dds://mars
       Default object: glb  default host: dds://mars
                           state: in service
       Checking clocks of glb replicas
       dds://mars    1987/04/09.17:09
       dds://pluto   1987/04/09.17:09
       dds://mercury1987/04/09.17:07
```

SEE ALSO

glbd(8), lb_admin(8)
Managing the NCS Location Broker.

NAME

dtcb – dump contents of tcp control blocks

SYNOPSIS

/etc/dtcb [–f] [[–t]
[<*tcb addr*>|–a] |–u
[<*ucb_addr*> | –a]]

DESCRIPTION

The command **dtcb** dumps the contents of the **tcp** control blocks associated with a particular tcp connection. The address of the **tcb** to be dumped may be obtained using the **netstat** program. Two control blocks are dumped: the **ucb** (user control block) which contains the send and receive queues and user-related flags, and the **tcb** (tcp control block) which contains the connection sequence numbers, state, flags, and out-of-sequence queues.

OPTIONS

–f	Force output if tcpd not running.
–t <*tcb_addr*>	Hexadecimal address of a **tcb** or, if not supplied, all **tcb**s.
–u <*ucb_addr*>	Hexadecimal address of a **ucb** or, if not supplied, all **ucb**s.
–a	All (both **tcb**'s and **ucb**'s for each socket).

EXAMPLES

The dump of a tcp control block for a listening ftp connection might look like this:

```
$ /etc/dtcb -t 1A9A50
ucb at 0x1A99C4:
local 0.0.0.0  lport 21
host  0.0.0.0  fport 0
uc_snd 8192  uc_ssize 0  uc_rcv 8192  uc_rsize 0
uc_shead 0  uc_stail 0  uc_rhead 0  uc_rtail 0
xflag:
UREUSEADDR UCANACCEPT
iostate:

status:
UCLOSED
flags:
UTCP
oobmark 0  oobcnt 0
```

```
TCB at 0x1A9A50:
lport 0x0  fport 0x0
t_state LISTEN
irs 00000000 rcv_urp 00000000 rcv_urg 00000000 rcv_nxt 00000000
    rcv_end 00000000
iss 1E3202D4 seq_fin 1E3202D4 snd_end 1E3202D4
      snd_urp 00000000 snd_lst 00000000

snd_nxt 1E3202D4 snd_una 1E3202D4 snd_wl 00000000
                                  snd_hi 1E3202D4

rex_val 00000000 rtl_val 00000000 xmt_val 00000000
flags:
snd_wnd 0  maxseg 0  xmtime 2  rxtct 0
timers:
INIT 0  REXMT 0  REXMTTL 0  PERSIST 0  FINACK 0
t_rcv_next 1A9A50  t_rcv_prev 1A9A50
```

NAME

edmtdesc – edit magtape descriptor file

SYNOPSIS

/etc/edmtdesc {*options*} *pathname*

DESCRIPTION

edmtdesc allows you to create, list, and modify magnetic tape descriptor objects. These descriptor files provide information to the streams manager so that it can handle subsequent tape operations much as an sio descriptor file describes the configuration of an sio line.

pathname (required) Specify name of magtape descriptor file to be created, listed, or edited.

OPTIONS

At least one of the following options must be specified.

–c Create a new magtape descriptor object with the name given in the *pathname* argument.

–l [*var...*] List the values of the variable(s) specified. If no variables are named, the entire magtape descriptor is listed.

–s {*var value*}... Set the variable(s) indicated to the specified value(s). At least one variable/value pair is required if –s is specified. Multiple variable/value pairs are permitted, separated by blanks.

VARIABLES

The variables known to **edmtdesc** are listed below, along with their types and default values. The variable types are: integer (int), Boolean (y/n), character string of *n* letters (c [*n*]), and date (in format yy/mm/dd.hh:mm).

Name	Type	Default	Definition
dev	c[1]	m	device type ('m' for magtape, 'c' for cartridge)
u	int	0	magtape unit number (normally 0)
lab	y/n	yes	'yes' if magtape is ANSI labeled, 'no' if unlabeled
reo	y/n	no	'yes' to reopen previously used volume, 'no' to open new volume ('yes' suppresses rewind)
clv	y/n	yes	'yes' closes volume when file is closed, 'no' leaves volume open

Name	Type	Default	Definition
spos	y/n	no	'yes' saves volume position when volume is closed (for reopen), 'no' rewinds volume when closed
vid	c[6]	-auto	volume identifier (labeled volumes)
vacc	c[1]		volume accessibility (labeled volumes)
own	c[14]	-auto	volume owner (labeled volumes)
f	int*	1	file sequence number: integer or "cur" for current file, or "end" for new file at end of labeled volume
rf	c[1]	D	record format -- "f" for fixed length, "d" for variable length, "s" for spanned, "u" for undefined
bl	int	2048	block length, in bytes
rl	int	2048	(maximum) record length, in bytes
ascnl	y/n	yes	'yes' for ascii newline handling (strip newlines on write, supply them on read), 'no' for no newline handling
fsect	int	1	file section number (labeled volumes)
fid	c[17]		file identifier (labeled volumes)
fsid	c[6]		file set identifier (labeled volumes)
gen	int	1	generation of file (labeled volumes)
genv	int	1	generation version of file (labeled volumes)
cdate	date	-auto	creation date of file (labeled volumes)
edate	date	-auto	expiration date of file (labeled volumes)
facc	c[1]		file accessibility (labeled volumes)
sysc	c[xx]		system code (labeled volumes)
sysu	c[xx]		system use (labeled volumes)
boff	int	0	buffer offset (labeled volumes, should be 0)

For cartridge tape (**dev c**), you must change the block length (**bl**) and the record length (**rl**) to be 512 or less and the record format to be fixed (**rf f**).

EXAMPLES

Edit file **set_tape**; set the tape unit number to 1; declare tape as ANSI labeled.

```
$ /etc/edmtdesc set_tape -s u 1 lab yes
```

Create descriptor file **ct** for cartridge tape, blocking 4 records of maximum length 128 to each block.

```
$ /etc/edmtdesc ct -c -s dev c bl 512 rl 128 rf f
```

NAME

edns – invoke editor for **ns_helper**

SYNOPSIS

/etc/edns [[*net.*]*node_id*]

DESCRIPTION

edns allows you to inspect and/or modify **ns_helper**'s master network root directory and replica list. Once invoked, **edns** enters an interactive mode and accepts the commands described in **help edns** commands.

[*net.*]*node_id* (optional) Set the default **ns_helper** to the **ns_helper** at the node specified by the internet address.

Default if omitted: Set the default **ns_helper** to any active **ns_helper**. An **ns_helper** becomes active after its database has been initialized.

NAME

/etc/edrgy – edit the network registry database

SYNOPSIS

/etc/edrgy [**–a** | **–p** | **–g** | **–o**] [**–l**] [**–s** *//site*] [**–synch**] [**–v**]

DESCRIPTION

The **edrgy** tool views and edits information in the registry database. You can invoke **edrgy** from any node.

Though anyone can read information in the registry database, you can usually change information only if you own the affected database entries. For example, only the owner of a group can add a name to the group's membership list.

With **edrgy**, you can edit and view names, accounts, and policies in the network registry, as well as entries in the local registry. The tool operates in one of four domains: person names, group names, organization names, and accounts.

OPTIONS

You can specify only one of **–a**, **–p**, **–g**, and **–o**.

–a (default)	Edit or view accounts.
–p	Edit or view persons.
–g	Edit or view groups.
–o	Edit or view organizations.
–l	Edit or view entries in local registry.
–s	Use the specified registry site.
–synch	Synchronize local registry with network registry.
–v	View selected entries.

Unless you specify the **–v** option, **edrgy** operates interactively. The following sections describes the commands you can enter in the interactive mode.

COMMANDS FOR PERSONS, GROUPS, AND ORGANIZATIONS

v[iew] [*name* | *number*] [**–f**] [**–m**] [**–po**]

View *name* entries.

If you specify a *number*, **edrgy** displays all matching entries, including any aliases.

The **–f** option displays entries in full (all fields except the membership list and organization policy).

If you are viewing groups or organizations, **–m** displays the membership list. For persons, **–m** lists all groups of which the person is a member, including groups that cannot appear in a project list.

If you specify **–po** while viewing organizations, **edrgy** displays policy information. Otherwise, it shows only the name and the UNIX number.

a[dd] [*person number* [*fullname*] [**–al**] [**–o** *owner*]]
a[dd] [*group number* [*fullname* [*password*]] [**–nl**] [**–o** *owner*]]
a[dd] [*organization number* [*fullname* [*password*]] [**–o** *owner*]]

Create a new name entry.

If you do not specify a *person*, *group*, or *organization* name, the **add** command enters an interactive mode and prompts you for each field in the entry. If you are adding organizations in the interactive mode, the command prompts you for policy information as well.

Specify the *owner* as a *person.group.organization* triplet. You can use % as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default owner, which you can set or display with the **defaults** command.

For persons, the **–al** option creates an alias entry. If *number* (the UNIX number) is already assigned to a person and you do not specify **–al**, an error occurs and you must either choose a different *number* or specify **–al**. If you use **–al** to create an alias and *number* is not already associated with a primary name, **edrgy** issues a warning but creates the alias.

For groups, the **–nl** flag indicates that the group is not to be included on project lists; omitting this flag allows the group to appear on project lists.

For groups and organizations, a space between quotation marks indicates a nil password.

Use quotation marks to embed spaces (or quotation marks) in a *fullname*. A single space between quotation marks indicates a nil *fullname*.

c[hange] [*person* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–al** | **–pr**]]
c[hange] [*group* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*] [**–nl** | **–l**]]
c[hange] [*organization* [**–n** *name*] [**–u** *number*] [**–f** *fullname*] [**–o** *owner*]
[**–p** *password*]]

Change a *name* entry.

If you do not specify a *person*, *group*, or *organization* name, the **change** command enters an interactive mode and prompts you for a name. If

you do not specify any fields, the command prompts you for each field in succession. To leave a field unchanged, press <RETURN> at the prompt. If you are changing organization entries in the interactive mode, the command prompts you for policy information as well.

For person entries, the **–al** flag changes a primary name into an alias, while the **–pr** flag changes an alias into a primary name. This change can be made only from the command line, not in the interactive mode.

For group entries, the **–nl** flag disallows the group from appearing in project lists, while the **–l** flag allows the group to appear in project lists.

For organization entries, you can change policy information only in the interactive mode.

A single space between quotation marks indicates a nil *fullname* or *password*.

Specify the *owner* as a *person.group.organization* triplet. You can use % as a wildcard for any or all of the components.

Changes to a person name are reflected in membership lists that contain the person name. For example, if the person **ludwig** is a member of the group **composers** and the person name is changed to **louis**, the membership list for **composers** is automatically changed to include **louis** but not **ludwig**.

Changes to *number* (the UNIX number) cause the operating system to change its mapping of the UID, the primary name, and any aliases from the old *number* to the new one. However, files owned by the old *number* do not automatically show the new *number* as their owner.

The only fields of reserved entries that you can change are the *fullname*, the *password*, the *owner*, and (for *groups*) the property that allows a *group* to appear in project lists. If a reserved *group* is allowed to appear in project lists, you can disallow it; but if the *group* is disallowed, you cannot allow it.

m[ember] [*group* | *organization* [**–a** *member_list*] [**–r** *member_list*]]

Edit the membership list for a group or organization.

If you do not specify a group or organization, the **member** command enters an interactive mode and prompts you for names to add or remove.

The **–a** flag precedes the person names (separated by spaces) to be added to the membership list, while the **–r** flag precedes those to be removed. If you do not include either flag on the command line, **edrgy** prompts you for names to add or remove.

Adding a person to a membership list permits creation of a login account for that person with that group or organization.

Removing *person* from the membership list for *group* has the side effect of deleting all login accounts of the form *person.group*, and likewise for organizations.

del[ete] { *person* | *group* | *organization* }

Delete a name entry.

You cannot delete reserved names. Deleting a group or organization has the side effect of deleting any accounts with that group or organization.

adopt *uid_high.uid_low person number* [*fullname*] [**–o** *owner*]
adopt *uid_high.uid_low group number* [*password* [*fullname*]] [**–nl**] [**–o** *owner*]
adopt *uid_high.uid_low organization number* [*password* [*fullname*]] [**–o** *owner*]

Create a primary name entry for the specified UID.

The UID must be an orphan (a UID for which no name exists in any domain). The *uid_high* and *uid_low* are hexadecimal numbers.

An error occurs if you specify a name or UNIX number that is already defined within the same domain of the database.

A single space between quotation marks indicates a nil *fullname* or *password*.

Specify the *owner* as a *person.group.organization* triplet. You can use % as a wildcard for any or all of the components. If you do not use the **–o** option, **edrgy** assigns the default *owner*, which you can set or display with the **defaults** command.

COMMANDS FOR ACCOUNTS

In all of the account operations, the *account* argument is a *person.group.organization* triplet such as **jones.graphics.research**. Unless otherwise specified, any or all of the components can be the wildcard character, %. For example, **view %.dev.%** views all accounts associated with the group **dev**.

In an *account* argument, if you omit a trailing *organization* (or *group.organization*), % (or %.%) is assumed. Thus, **keats.%.%**, **keats.%**, and **keats** are equivalent.

v[iew] [*account*] [**–f**]

Display login accounts specified by the *account* pgo (*person, group, organization*) triplet.

Without the **–f** flag, **view** displays only the user fields in each account entry: abbreviated account S encrypted password, miscellaneous information, home directory, and login shell.

With **–f**, **view** displays the full entry, including the administrative fields as well as the user fields. Administrative information includes who created the account, when it was created, who last changed it, when it was last changed, when it expires, whether it is valid, whether the password is valid, and when the password was last changed.

a[dd] [*account* [**–a** { **p** | **pg** | **pgo** }] [*password* [*misc* [*homedir* [*shell*]]]]
[**–pnv**] [**–x** *account_exp* | **none**] [**–anv**]]

Create a login account.

Specify *account* as a pgo triplet. Wildcards are not allowed. If you do not supply an *account* on the command line, **add** enters an interactive mode and prompts you for each field in succession.

If the person specified in *account* is not already a member of the specified group and/or organization, **edrgy** automatically attempts to add the person to the membership lists. If you are not an owner of the group and/or organization, the attempt will fail and the account will not be created.

The **–a** flag indicates the degree of abbreviation allowed for login: **p** means that only the person is required; **pg** means the person and the group; **pgo** means that all three components of the account SID are required. (Of course, a user can always supply more components than are required.) If the abbreviation you specify is already defined for another account, **edrgy** automatically uses the shortest unique abbreviation and issues a warning.

For example, if you create an account **babar.elephants.none** with the abbreviation **p**, a user need only enter **babar** at the login prompt to use the account. If you then create an account **babar.kings.none**, the **p** abbreviation will conflict with the existing account, so the **pg** abbreviation, **babar.kings**, will be the shortest unique one.

Omitting the **–a** is equivalent to specifying **–a p** and results in use of the shortest unique abbreviation.

The *password* must adhere to the policy of the associated organization or the policy of the registry as a whole, whichever is more restrictive.

The *misc* field is not used by the operating system. The *gecos* field of each account's entry in the /**etc/passwd** file is the concatenation of the person's full name and the account's *misc*. Use quotes to include spaces, hyphens, or quotes in *misc*.

The *homedir* and *shell* are pathnames. The default *homedir* is /. The default *shell* is the null string.

Use a single space between quotation marks to indicate a nil *password*, *misc_info*, *homedir*, or *shell*.

The **−pnv** (password not valid) flag specifies that at the next login (for a newly created account, the first login), the user must change the password. If you omit this option, the password is valid.

The **−x** flag sets an expiration date for the account; the default is none.

The **−anv** (account not valid) flag specifies that the account is not currently valid for login. If you omit this option, the account is valid.

c[hange] [*account* [**−n** *new_account*] [**−a** { **p** | **pg** | **pgo** }]
[**−p** *password*] [**−m** *misc*] [**−h** *homedir*] [**−s** *shell*]
[**−pnv** | **−pv**] [**−x** *account_exp* | **none**] [**−anv** | **−av**]

Change one or more account entries.

Specify *account* as a pgo triplet. Wildcards are allowed, unless you use the **−n** option. If you do not supply an *account* on the command line, **change** enters an interactive mode and prompts you for each field in succession. Press <RETURN> to leave a field unchanged.

The command line arguments are largely the same as those of the **add** command. The **−n** flag enables you to change the account SID to *new_account*, a pgo triplet that cannot contain wildcards. The **−pv** flag specifies that the password is valid. The **−av** flag specifies that the account is valid.

You can enter a single space between quotation marks to indicate a nil *password*, *misc*, *homedir* or *shell*.

del[ete] *account*

Delete the entry for *account*, a pgo triplet that cannot contain wildcards.

MISCELLANEOUS COMMANDS

do[main] [**p** | **g** | **o** | **a**]

Change or display the type of registry information being viewed or edited.

You can specify **p** for persons, **g** for groups, **o** for organizations, or **a** for accounts. If you supply no argument, **edrgy** displays the current domain.

s[ite] [*//site*] [**–l**]

Change or display the registry site being viewed or edited.

If you specify a *//site*, **edrgy** attempts to use the registry server at the named site. If you specify **-l**, **edrgy** uses the local registry. If you supply no argument, **edrgy** displays the current site.

prop[erties]

Change and/or display the registry properties and policies.

This command prompts you for any changes to make. Press <RETURN> to leave information unchanged.

synch[ronize]

Update the local registry to match the master registry.

If a matching entry cannot be retrieved from the network registry, the local entry is marked invalid for login, and its UNIX numbers are updated.

co[py] [*account*]

Copy information for the specified accounts from the master registry to the local registry.

The *account* is a pgo triplet that can contain wildcards; trailing wildcard components can be omitted. If a matching account already exists in the local registry, **edrgy** updates the information to match that in the master registry; otherwise, **edrgy** adds the entry. If all entries in the local registry are used, **copy** reports an error and terminates.

def[aults]

Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
Display usage information for **edrgy**.

If you do not specify a particular command, **edrgy** lists the available commands.

q[uit] Exit **edrgy**.

COMMANDS VALID FOR THE LOCAL REGISTRY

To edit or view the local registry, use the **–l** flag when you invoke **edrgy**. This section lists the commands that are valid for editing or viewing the local registry. Unless otherwise specified, all options are as described in the previous command descriptions.

v[iew] [*name* | *number*] [**–f**] [**–po**]
View name entries. (The **–m** option is not valid.)

v[iew] [*account*] [**–f**]
Display specified login accounts.

c[hange] [*account* [**–a** { **p** | **pg** | **pgo** }] [**–m** *misc*] [**–h** *homedir*] [**–anv**]
Change one or more account entries. (The **–p**, **–s**, **–pnv**, **–pv**, **–x**, and **–av** options are not valid.)

del[ete] *account*
Delete an account entry.

do[main] [**p** | **g** | **o** | **a**]
Change or display the type of registry information being viewed or edited.

s[ite] [*//site*] [**–l**]
Change or display the registry site being viewed or edited.

prop[erties]
Change and/or display the registry properties and policies.

synch[ronize]
Update the local registry to match the master registry.

co[py] [*account*]
Copy information for the specified accounts from the master registry to the local registry.

def[aults]
Change and/or display the default values that **edrgy** uses.

h[elp] [*command*]
Display usage information for **edrgy**.

q[uit] Exit **edrgy**.

NAME

environment – inquire about system environment

SYNOPSIS

/etc/environment [–c] [–i]

DESCRIPTION

This command is used by shell scripts to inquire about the current "environment", installed environments, or both.

OPTIONS

If no flags are specified, the current environment is printed. If **–i** is specified, however, and you also want current environment, you must add **–c**.

–c print current environment

–i print installed environments

EXAMPLES

```
$ /etc/environment
aegis

$ /etc/environment –c
aegis

$ /etc/environment –i
aegis sysv bsd
```

NAME

find_orphans – locate and catalog uncataloged objects

SYNOPSIS

/etc/**find_orphans** [*options*] [*volume_pathname*]

DESCRIPTION

find_orphans finds all uncataloged permanent objects in a local volume. It uses or creates a directory "lost+found" in the root of the volume and creates entries for all objects not cataloged elsewhere.

find_orphans can operate by itself by using search mode, in which case it searches the volume for all orphans, or it works with **salvol** (list mode, the default), in which case it just catalogs the orphans detected by the previous run of **salvol**. Both methods find exactly the same set of orphans. We recommend that you run **find_orphans** every time a node is booted, and on every mounted volume. Below is a description of the two modes of operation.

The objects cataloged by **find_orphans** are given sequential names like f1, f2, etc., and you can move them using **/com/mvf** or **/bin/mv** to a directory of your choice. **find_orphans** is useful for finding objects that were being updated during a system crash or that were uncataloged through program errors.

In list mode (the default), **find_orphans** catalogs all objects listed in the file **lost+found.list** in the root directory of the volume. If the file does not exist, **find_orphans** creates it. Note that **invol** creates the file when it creates the volume. If the **lost+found.list** exists, **salvol** enters information describing each orphan. List mode is considerably faster than search mode since there is no need to search the entire volume. You must have permission to catalog objects in **lost+found**.

In search mode, you must have read permission to all directories on the volume. If some directory is not readable, every object under that directory is cataloged in the **lost+found** directory. In addition, you must have permission either to create the **lost+found** directory or to catalog objects in **lost+found** when it already exists.

Search mode should be run only on a quiescent node; that is, one not connected to the network (use **netsvc –n** to disable network communications) and not actively running any processes other than the one performing the **find_orphans** operation.

volume_pathname (optional)
Specify the name of the volume to be searched. The volume must be physically attached to your node; you may not find orphan objects on volumes elsewhere in the network.

Default if omitted: search node boot volume

OPTIONS

–l[ist] (default)	List mode, catalog objects listed in **/lost+found.list**.
–s[earch]	Search mode, search the volume for orphans.
–v[erify]	Verify only; don't catalog any orphans.

EXAMPLES

```
$ /etc/find_orphans
Cataloging in: /lost+found
39D40FF0.100086CA  ->  f1
39D40F9D.E00086CA  ->  f2
39D41026.600086CA  ->  f3
39D40DA6.D00086CA  ->  f4
39D40998.200086CA  ->  f5
39D41042.800086CA  ->  f6
39D40CB8.E00086CA  ->  f7
39D41001.300086CA  ->  f8
39D40F7E.D00086CA  ->  f9
39D40CCE.F00086CA  ->  f10
39D40D8B.C00086CA  ->  f11
39D40E33.100086CA  ->  f12
39D40A06.700086CA  ->  f13
39D40F23.900086CA  ->  f14
39D40E16.000086CA  ->  f15
39D40F36.A00086CA  ->  f16
39D41C0A.200086CA  ->  f17
Number of orphans catalogued: 17
```

$ ld –a /lost+found

```
Directory "/lost+found":

sys   type      blocks  current
type  uid         used   length  attr rights       name

file  uasc           1       32  P    prwx-        f1
file  unstruct       0        0  P    prwx-        f10
file  uasc           1       32  P    prwx-        f11
file  unstruct       0        0  P    prwx-        f12
file  unstruct       1       54  P    prwx-        f13
file  uasc           1       32  P    prwx-        f14
file  uasc           1       32  P    prwx-        f15
file  unstruct       0        0  P    prwx-        f16
file  unstruct       0        0  P    prwx-        f17
file  unstruct       0        0  P    prwx-        f2
file  uasc           1       32  P    prwx-        f3
file  unstruct       0        0  P    prwx-        f4
file  coff           1   101376  P    prwx-        f5
file  unstruct       0        0  P    prwx-        f6
file  uasc           1       32  P    prwx-        f7
file  unstruct       0        0  P    prwx-        f8
file  uasc           1       32  P    prwx-        f9

17 entries, 9 blocks used.
```

NAME

ftpd – DARPA Internet File Transfer Protocol server

SYNOPSIS

/etc/ftpd [**–d**] [**–l**] [–t*timeout*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification; see **services**(5).

OPTIONS

–d	Write debugging information to the syslog.
–l	Log each **ftp** session in the syslog.
–t*timeout*	Set the inactivity timeout period to *timeout*. Without this option, the **ftp** server will timeout an inactive session after 15 minutes.

FTP REQUESTS

The **ftp** server currently supports the following **ftp** requests; case is not distinguished.

Request	Description
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	List files in a directory ("**ls -lg**")
MKD	Make a directory
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory ("**ls**")
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from filename
RNTO	Specify rename-to filename
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer *structure*

TYPE	Specify data transfer *type*
USER	Specify username
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The remaining **ftp** requests specified in Internet RFC 959 are recognized, but not implemented.

The **ftp** server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959.

ftpd interprets filenames according to the ''globbing'' conventions used by **csh**(1). This allows users to utilize the metacharacters ''*?[]{}~''.

ftpd authenticates users according to four rules.

1) The username must be in the password data base, **/etc/passwd**, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2) The username must not appear in the file **/etc/ftpusers**.

3) The user must have a standard shell returned by **getusershell**(3).

4) If the username is ''anonymous'' or ''ftp'', an anonymous ftp account must be present in the password file (user ''ftp''). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot**(2) command to the home directory of the ''ftp'' user. In order that system security is not breached, we recommend that the ''ftp'' subtree be constructed with care; the following rules are recommended.

~ftp	Make the home directory owned by ''ftp'' and unwritable by anyone.
~ftp/bin	Make this directory owned by the super-user and unwritable by anyone. The program **ls**(1) must be present to support the list commands. This program should have mode 111.
~ftp/etc	Make this directory owned by the super-user and unwritable by anyone. The files **passwd**(5) and **group**(5) must be present for the **ls** command to work properly. These files should be mode 444.
~ftp/pub	Make this directory mode 777 and owned by ''ftp''. Users should then place files which are to be accessible via the anonymous account in this directory.

CAUTIONS

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

NAME

gettable – get NIC format host tables from a host

SYNOPSIS

/etc/gettable [–v] *host* [*outfile*]

DESCRIPTION

gettable is a simple program used to obtain the NIC standard host tables from a "nicname" server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file *outfile* or by default, **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for "nicname". A request is then made for "ALL" names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable**(8) program which converts the NIC standard file format to that used by the network library lookup routines.

OPTIONS

–v	Get just the version number instead of the complete host table, and puts the output in the file *outfile* or by default, **hosts.ver**.
outfile	Place the tables in the specified *outfile*.

CAUTIONS

If the name-domain system provided network name mapping well as host name mapping, **gettable** would no longer be needed.

SEE ALSO

intro(3N), htable(8), named(8);
Configuring and Managing TCP/IP.

NAME

getty – set terminal mode

SYNOPSIS

/etc/getty [*type* [*tty*]]

DESCRIPTION

getty is usually invoked by **init**(8) to open and initialize the **tty** line, read a login name, and invoke **login**(1). **getty** attempts to adapt the system to the speed and type of terminal being used.

The argument *tty* is the special device file in **/dev** to open for the terminal (for example, **ttyh0**). If there is no argument or the argument is a dash (–), the **tty** line is assumed to be open as file descriptor 0.

The *type* argument can be used to make **getty** treat the terminal line specially. This argument is used as an index into the **gettytab**(5) database, to determine the characteristics of the line. If there is no argument, or there is no such table, the default table is used. If there is no **/etc/gettytab** a set of system defaults is used. If indicated by the table located, **getty** will clear the terminal screen, print a banner heading, and prompt for a login name. Usually either the banner or the login prompt will include the system hostname. Then the user's name is read, one character at a time. If a null character is received, it is assumed to be the result of the user typing the break (interrupt) character. The speed is usually then changed and the login prompt is typed again; a second "break" changes the speed again and the login prompt is typed once more. Successive break characters cycle through the same standard set of speeds.

The user's name is terminated by a newline or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see **tty**(4)).

The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is nonempty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

Finally, **login** is called with the user's name as an argument.

Most of the default actions of **getty** can be circumvented, or modified, by a suitable **gettytab** table.

getty can be set to timeout after some interval, which will cause dial up lines to hang up if the login name is not entered reasonably quickly.

DIAGNOSTICS

"ttyxx: No such device or address." A terminal which is turned on in the **ttys** file cannot be opened, probably because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in **/dev** does not exist.

FILES

/etc/gettytab

SEE ALSO

gettytab(5), init(8), login(1), ioctl(2), tty(4), ttys(5);
Using Your BSD Environment.

NAME

glbd – Global Location Broker Daemon

SYNOPSIS

/etc/ncs/glbd [–create { –first | –from *host_name* **}]**

DESCRIPTION

The Global Location Broker Daemon (**glbd**), part of the Network Computing System (NCS), manages the Global Location Broker (GLB) database. The GLB database stores the locations of NCS-based server programs on a network or internet.

The GLB can be replicated for greater availability of its database. Copies of the database can exist on several nodes, with the brokers maintaining consistency of the replicated database. (In an internet, at least one **glbd** must run in each network.) The **drm_admin** tool administers the replication of the GLB database.

Access to the GLB database by clients is supported on both the DARPA IP and the Domain DDS network protocols. However, GLBs use only the DDS protocols to maintain replication of the database. Thus, on an internet, all routing nodes must support DDS.

A Local Location Broker Daemon (**llbd**) must be running on the local node when **glbd** is started. Typically, both brokers are started at boot time from the **/etc/rc** file.

If **glbd** is to communicate via IP protocols, a TCP daemon (**tcpd**) must also be running on the local node. **tcpd** should be started before **llbd**.

See *Managing the NCS Location Broker* for more information.

Diagnostic output from **glbd** is written to the file `**node_data/system_logs/glb_log**.

OPTIONS

–create Create a replica of the GLB. This option creates a GLB database in addition to starting a broker process. It must be used with either **–first** or **–from**.

–first This option must be used with the **–create** option. Use it to create the first replica (i.e., the very first instance) of the GLB on your network or internet.

–from *host_name*

This option must be used with the **–create** option. Use it to create additional replicas of the GLB. A replica of the GLB must exist at *host_name*. The database and replica list for the new replica are initialized from those at *host_name*. The replica at *host_name* adds an entry for the new replica to its replica list and propagates the entry to the other GLB replicas.

A *host_name* takes the form *family*:*host*. The only currently supported family is **dds**; a host in this family is specified by its entry directory or its network address. For example, **dds://jeeves** and **dds:#1234.abcd** are acceptable host names.

EXAMPLES

Initialize and start the first replica of the GLB on this network or internet:

```
$ /etc/server /etc/ncs/glbd –create –first &
```

Start a subsequent replica of the GLB, initializing its database from host //jeeves:

```
$ /etc/server /etc/ncs/glbd –create –from dds://jeeves &
```

Restart an existing replica of the GLB:

```
$ /etc/server /etc/ncs/glbd &
```

Restart an existing replica of the GLB on remote host //bertie:

```
$ crp –on //bertie /etc/server //bertie/etc/ncs/glbd &
```

SEE ALSO

drm_admin(8), lb_admin(8), llbd(8), *Managing the NCS Location Broker*.

NAME

halt – stop the processor

SYNOPSIS

/etc/halt [–n] [–q] [–y]

DESCRIPTION

The **halt** command writes out cached pages to the disks and then stops the processor. The machine does not reboot.

halt normally logs the shutdown using syslog(8) and places a shutdown record in the login accounting file **/usr/adm/wtmp.** These actions are inhibited if the **–n** or **–q** options are present.

OPTIONS

–n	Prevent the sync before stopping.
–q	Cause a quick halt, without attempting a graceful shutdown.
–y	Halt the system from a dialup.

SEE ALSO

reboot(8), shutdown(8), syslogd(8)

NAME

hostns – convert host table files to resource record format

SYNOPSIS

hostns [**–f** *file*] [**–d** *domain*] [**–h** *host*] [**–n** *subnet_id*] [**–s** *major.minor*] [**–u** *user*]

DESCRIPTION

The **hostns** command converts host address information obtained from an existing /**etc/hosts** format file (see **hosts**(5)) into Standard Resource Record Format for use by **named**(8). It is intended to serve as an aid in bringing up the name server on an existing network. By default, **hostns** uses host address mapping information contained in the file /**etc/hosts.**

OPTIONS

–f *file* — Specify an alternate host address mapping file, named *file*.

–d *domain* — Use *domain* as the domain name for the local zone. Normally if the current machine's hostname contains any dots, the portion of the hostname after the first dot is taken to be the domain name for the local zone. If there are no dots in the hostname, the domain name defaults to .MY.DOMAIN.

–h *host* — Build files for a different host. Normally **hostns** assumes **named**(8) will be run on the local machine and generates nameserver records.

–n *subnet_id* — Limit the conversion to the specific subnet. Normally, **hostns** converts all of the entries in the specified /**etc/hosts** file. *subnet_id* may be specified in hexadecimal or in the standard Internet "dotted address" format. The **–n** option may be specified, with a different *subnet_id*, up to five times.

–s *major.minor* — Set the serial number in the Start of Authority (SOA) record to *major.minor* instead of 1.1. The serial number is used to detect updates in a running network.

–u *user* — Specify *user* as the responsible administrator. Normally, **hostns** assumes the user responsible for network administration is the user running the program.

FILES

The following files are produced:

named.boot	boot file
named.ca	initial cache data
named.local	localhost reverse pointer
named.hosts	hosts file
named.rev	hosts file reverse pointers

SEE ALSO

hosts(5), named(8), nshost(8);
Configuring and Managing TCP/IP.

NAME

htable – convert NIC standard format host tables

SYNOPSIS

/etc/htable [–c *connected-nets*] [–l *local-nets*] *file*

DESCRIPTION

The **htable** command is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running **htable**: **hosts**, **networks**, and **gateways**. The **hosts** file may be used by the **gethostbyname**(3N) routines in mapping host names to addresses if the nameserver, **named**(8), is not used. The **networks** file is used by the **getnetent**(3N) routines in mapping network names to numbers. The **gateways** file may be used by the routing daemon in identifying ''passive'' Internet gateways; see **routed**(8c) for an explanation.

If any of the files **localhosts**, **localnetworks**, or **localgateways** are present in the current directory, the file's contents are prepended to the output file. Of these, only the gateways file is interpreted. This feature allows sites to maintain local aliases and entries which are not normally present in the master database. Only one gateway to each network will be placed in the gateways file; a gateway listed in the **localgateways** file will override any in the input file.

OPTIONS

–c *connected-nets*

If the gateways file is to be used, use this flag to specify a list of networks to which the host is directly connected. The networks, separated by commas, may be given by name or in Internet-standard dot notation, for example,

–c arpanet,128.32,local-ether-net.

htable only includes gateways that are directly connected to one of the specified networks, or that can be reached from another gateway on a connected net.

–l *local-nets* The argument is a comma-separated list of networks to be treated as ''local''. List format rules are the same as for –c. Information about hosts on local networks is taken only from the **localhosts** file. Entries for local hosts from the main database will be omitted. This allows the **localhosts** file to completely override any entries in the input file.

htable is best used in conjunction with the **gettable**(8C) program which retrieves the NIC database from a host. If the name-domain system provided network name mapping well as host name mapping, **htable** would no longer be needed.

SEE ALSO

intro(3N), gettable(8C), named(8);
Configuring and Managing TCP/IP.

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/etc/ifconfig *interface [address_family*] [*address* [*dest_address*]] [*parameters*]
/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine. It may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form *name unit*, for example, **eth0**.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, it is necessary to specify the *address_family*, which may change the interpretation of the remaining parameters. The only address family currently supported by Apollo is **inet .**

For the DARPA-Internet family, the address is either a host name present in the host name data base, **hosts**(5), or a DARPA Internet address expressed in the Internet standard ''dot notation''.

PARAMETERS

The following parameters may be set with **ifconfig**:

up Mark an interface ''up''. This may be used to enable an interface after an ''ifconfig down.'' It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down Mark an interface ''down''. When an interface is marked ''down'', the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers Request the use of a ''trailer'' link level encapsulation when sending (default). If a network interface supports **trailers**, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see **arp**(4P); currently, only 10 MB ETHERNET), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.

–trailers Disable the use of a "trailer" link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10MB ETHERNET addresses.

–arp Disable the use of the Address Resolution Protocol.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**(8c)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

debug Enable driver dependent debugging code; usually, this turns on extra console error logging.

–debug Disable driver dependent debugging code.

netmask *mask* (Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table **networks**(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dstaddr Specify the address of the correspondent on the other end of a point to point link.

broadcast (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

intro(4N), netstat(1), rc(8);
Configuring and Managing TCP/IP.

NAME

import_passwd – create registry entries based on information in UNIX group and password files

SYNOPSIS

/etc/import_passwd [**–i**] [**–a** | **–f**] [**–c**] [**–o** *org*] **–s** *pathname* [**–v**]

DESCRIPTION

import_passwd is a mechanism for creating Apollo registry entries that are consistent with foreign password and group file entries. You should use **import_passwd** to ensure consistency between Apollo and foreign protection mechanisms when you

- Attach Apollo node(s) to an existing UNIX network
- Attach UNIX node(s) to an Apollo network
- Connect Apollo and UNIX networks

If the foreign password and group file entries do not exist in the Apollo registry, **import_passwd** will create them. If there are duplicate entries, **import_passwd** will follow your directions on how to handle them. (Note that reserved names and reserved UNIX IDs cannot be reassigned.)

The Process

The Apollo registry must exist before you can use **import_passwd.** If you are simply adding a few Apollo nodes to a foreign network, you can create a new, but empty, registry to meet this requirement. Once the registry exists, the registry server must be running, and you must be logged on as root.

As **import_passwd** processes, it

1. Examines the foreign group file and creates group entries in the registry.
2. Examines the foreign passwd file and creates person, organization, and account entries in the registry. The organization assigned is specified as input to **import_passwd.**
3. Reexamines the foreign group file and creates membership lists.

Conflicts

During this process, **import_password** may find conflicts in name strings (for example, in the foreign network, joe 102; in Apollo, joe 555) and in UNIX IDs (for example, in the foreign network, joe 102; in Apollo, ann 102). **import_passwd** provides a number of options to help resolve these conflicts.

The Favored Entry

The **–a** (favor Apollo entry) or **–f** (favor foreign entry) options specify which entry should be favored. A favored entry is retained as is. You are prompted to modify non-favored entries. (Note, however, that in some cases you may be prompted to modify a favored entry. For example, if the non-favored entry is a reserved name, you will be prompted to modify the favored entry.)

Name Conflicts

The **–i** option specifies that duplicate names are not in conflict but in fact, represent the same identity. Therefore, when duplicate names arise, no action is necessary. If you do not use the **–i** option, **import_passwd** resolves the name conflict by prompting for a name string for the non-favored entry.

UNIX ID Conflicts

The resolution of UNIX ID conflicts is also determined by the favored entry. If a conflict exits, you are prompted for a new UNIX ID for the non-favored entry.

Other Registry Entries

Except for names and UNIX IDs, all other information stored in the Apollo registry for an existing identity is retained.

New registry entries created by **import_passwd** are assigned the following values:

For Person and Group Entries:

fullname = " (empty)

owner = Same as the owner of the organization specified with the **–o** option. If no organization is specified, then the owner of the organization named "none".

alias/primary = Primary for first entry; alias for subsequent ones.

projlist_ok = Yes.

passwd = For groups only, taken from the group file.

membership list = For new groups only, all persons listed in the group file, and all persons with accounts in the password file with that group.

For Account Entries:

abbreviation = Shortest possible abbreviation that does not conflict with pre-existing Apollo accounts.

acount_valid = True.

gecos = Same as UNIX password file.

homedir = Same as UNIX password file.

shell = Same as UNIX password file.

passwd = Same as UNIX password file. Note that you must modify or reset imported passwords before user authentication is possible and for the account to be usable in a pre-SR10 registry.

passwd_dtm = Date and time **import_passwd** was run.

passwd_valid = True.

OPTIONS

–i	Name strings are not in conflict, but represent the same identity.
–a (default)	Favor Apollo entries for conflicts.
–f	Favor foreign entries for conflicts.
–c	Run in check mode: Process the command, showing all conflicts, but make no requests for resolution.
–o *org*	*org* is the name of an Apollo organization to be assigned to all imported entries.
–s *pathname*	*pathname* is the path to the directory containing the foreign password and group files to be imported.
–v	Run in verbose mode: Generate a verbose transcript of. **import_passwd** activity.

SEE ALSO

edrgy(8), rgy_admin(8), rgy_merge(8), rgyd(8)

NAME

inetd – internet ''super–server''

SYNOPSIS

/etc/inetd [**–d**] [*configuration file*]

DESCRIPTION

The **inetd** server should be run at boot time by **/etc/rc.local**. It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases which will be described below). Essentially, **inetd** allows running one daemon to invoke several others, reducing load on the system.

Upon execution, **inetd** reads its configuration information from a configuration file which, by default, is **/etc/inetd.conf**. There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a ''#'' at the beginning of a line. The fields of the configuration file are as follows:

service name
socket type
protocol
wait/nowait
user
server program
server program arguments

The *service name* entry is the name of a valid service in the file **/etc/services/**. For ''internal'' services (discussed below), the service name *must* be the official name of the service (that is, the first entry in **/etc/services**).

The *socket type* should be one of **stream**, **dgram**, **qraw**, **rdm**, or **seqpacket**, depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The *protocol* must be a valid protocol as given in **/etc/protocols**. Examples might be **tcp** or **udp**.

The *wait/nowait* entry is applicable to datagram sockets only (other sockets should have a *nowait* entry in this space). If a datagram server connects to its peer, freeing the socket so **inetd** can received further messages on the socket, it is said to be a ''multi-threaded'' server, and should use the *nowait* entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be ''single-threaded'' and should use a *wait* entry. **comsat** (**biff**) and **talk** are both examples of the latter type of datagram server. **tftpd** is an exception; it is a datagram server that establishes pseudo-connections. It must be listed as *wait* in order to avoid a race; the server reads the first packet, creates a new socket, and then forks and exits to allow **inetd** to check for new service requests to spawn new servers.

The *user* entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root. The *server program* entry should contain the pathname of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be **internal**.

The arguments to the server program should be just as they normally are, starting with argv[0], which is the name of the program. If the service is provided internally, the word **internal** should take the place of this entry.

inetd provides several ''trivial'' services internally by use of routines within itself. These services are **echo, discard, chargen** (character generator), **daytime** (human readable time), and **time** (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are tcp based. For details of these services, consult the appropriate RFC from the Network Information Center.

inetd rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread.

SEE ALSO

comsat(8C), ftpd(8C), rexecd(8C), rlogind(8C), rshd(8C), telnetd(8C), tftpd(8C); *Configuring and Managing TCP/IP*.

NAME

init – process control initialization

SYNOPSIS

/etc/init

DESCRIPTION

init is invoked as the last step in the boot procedure. It normally runs the automatic reboot sequence as described in **reboot**(8), and if this succeeds, begins multi-user operation.

In multi-user operation, **init**'s role is to create a process for each terminal port on which a user may log in. To begin such operations, it reads the file **/etc/ttys** and executes a command for each terminal specified in the file. This command will usually be **/etc/getty**. **getty** opens and initializes the terminal line, reads the user's name and invokes **login** to log in the user and execute the shell.

Ultimately the shell will terminate because of an end-of-file either typed explicitly or generated as a result of hanging up. The main path of **init**, which has been waiting for such an event, wakes up and removes the appropriate entry from the file **utmp**, which records current users, and makes an entry in **/usr/adm/wtmp**, which maintains a history of logins and logouts. The **wtmp** entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and **getty** is reinvoked.

init catches the *hangup* signal (signal SIGHUP) and interprets it to mean that the file **/etc/ttys** should be read again. The shell process on each line which used to be active in **ttys** but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add terminal lines without rebooting the system by changing the **ttys** file and sending a *hangup* signal to the **init** process: use **kill –HUP 1**.

init will terminate multi-user operations and resume single-user mode if sent a terminate (TERM) signal, that is, **kill –TERM 1**. If there are processes outstanding which are deadlocked (due to hardware or software failure), **init** will not wait for them all to die (which might take forever), but will time out after 30 seconds and print a warning message.

init will cease creating new **getty**'s and allow the system to slowly die away, if it is sent a terminal stop (TSTP) signal, i.e., **kill –TSTP 1**. A later hangup will resume full multi-user operations, or a terminate will initiate a single user shell. This hook is used by **reboot**(8) and **halt**(8).

init's role is so critical that if it dies, the system will reboot itself automatically.

DIAGNOSTICS

/etc/getty *gettyargs* `failing, sleeping.`

A process being started to service a line is exiting quickly each time it is started. This is often caused by a ringing or noisy terminal line. **init** will sleep for 30 seconds, then continue trying to start the process.

```
WARNING: Something is hung (won't die); ps axl advised
```

A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

FILES

/dev/console
/dev/tty*
/etc/utmp
/usr/adm/wtmp
/etc/ttys
/etc/rc

SEE ALSO

login(1), kill(1), sh(1), ttys(5), getty(8), rc(8), reboot(8), halt(8), shutdown(8)

NAME

invol – initialize disk volumes

SYNOPSIS

from shell: /etc/**invol**
from Mnemonic Debugger: ex **invol**

DESCRIPTION

invol initializes physical disk volumes, creates logical volumes, and maintains badspot lists. Once initialized, a volume can be mounted with the **mount**(8) command, or can be used to bootstrap the operating system, providing it contains the necessary files. **invol** prompts for all required information.

SUMMARY OF OPTIONS (Complete description follows.)

0	Exit.
1 [–fnb5uom]	Initialize virgin physical volume.
2 [–fnb5u]	Add a logical volume.
3 [–fnb5]	Re-initialize an existing logical volume.
4	Delete a logical volume.
5	List logical volumes.
6	List badspots on disk or volume.
7	Create physical badspot list.
8	Create or modify a Domain/OS paging file.
9	Add to existing badspot list.
10	Display/change sector interleave factor.
11	Remove from existing badspot list.

FLAGS SUMMARY

–u	Use defaults
–f	Don't re-format disk
–o	Pre-SR10 format
–n	Make non-bootable volume
–b	Apply BSD UNIX ACLs
–5	Apply SysV UNIX ACLs
–m	Make a multi-disk set

FULL DESCRIPTIONS OF OPERATIONS

0 Exit

1 [–fnb5uom]

Initialize a virgin physical volume.

Every new disk must be initialized before it can be used. When you initialize a new disk, all existing data on the disk are overwritten. Do not initialize a disk that contains any data you need to save. We initialize Winchester disks during the manufacturing process, before installing the system software.

To initialize a new disk, follow this procedure:

A. **invol** asks which option to perform. Type **7** to create or replace the bad-spot list. (See "Recording Badspot Information"). Type **9** if you want to add to the existing badspot list. Otherwise, type **1** to initialize a new physical volume.

B. Specify the type of disk to initialize. **invol** prompts with:

```
Select disk: [w=Winch|s=Storage mod|f=Floppy|q=Quit]
             [ctrl#:][unit#]
```

Typing **q** (as always) will exit the program.

invol (as well as **salvol** and **fixvol**) can deal with multiple controllers of the same type. The encoding of the controller# and unit# is as follows:

```
w          Winchester ... Controller #0 ... Unit #0
w1         Winchester ... Controller #0 ... Unit #1
w1:        Winchester ... Controller #1 ... Unit #0
w1:2       Winchester ... Controller #1 ... Unit #2
```

Thus a single character N following the s|w specifies a unit# on the first (0'th) controller. If this character is followed by a ':' character, it is taken to be a controller specifier which is then followed by an optional unit specifier.

C. **invol** asks for the name of the physical volume.

D. Choose one of the following verification options:
   1 - No verification
   2 - Write all blocks on the volume
   3 - Write and reread all blocks on the volume

If you choose no verification (option 1), **invol** does not read or write to the disk, except to create the volume structure. This option is the fastest, but means that the disk is not verified until it is mounted and read or written.

If you choose the second option, **invol** attempts to write to each block on the disk. The third option, writing and rereading all blocks on the volume, is the safest but also the slowest. For example, to format a complete 33MB Winchester, option 1 requires about five minutes, option 2 requires about fifteen minutes, and option 3 requires about 30 minutes.

If a floppy disk is initialized with **invol** on a busy node, there is a small chance that a format operation will fail, but that the failure will not be reported to **invol**. For this reason, **invol** writes each block during floppy initialization, even for verification option 1. If a write fails after an apparently successful format, **invol** will print the message:

```
format failed for daddr <disk_address>:<write status>
- - retrying format
```

and will reformat (and rewrite) the track in error. This happens whether or not the floppy has been previously initialized.

E. Enter the average file size, when prompted:

```
Expected average file size, in blocks
(CR for default-5 blocks):
```

Press <RETURN> to accept the default value of 5 blocks. Supplying a relatively accurate value for the average file size can save space on the disk, because the volume table of contents (a system table) will be allocated more efficiently.

F. **invol** requests the size (in blocks) and name of each logical volume to be created. After each entry, **invol** tells you how much space remains. After entering the size and name of all logical volumes, enter a blank line to terminate input. A physical volume can contain up to 10 logical volumes. For example:

```
There are 1231 blocks available.

volume 1:  1231,vol1
```

The logical volume size must be at least 30 blocks, and must be a multiple of the track size for the disk. If you specify a logical volume size

that is not a multiple of the track size, **invol** rounds it up to the next multiple track size, and informs you. Note that the physical volume label occupies the first block on the volume. Thus, the size of the first logical volume is always one less than a multiple of the track size.

Logical volume names are optional, and are used only when **invol** lists the logical volumes on the disk (option 5). You cannot change the name of a logical volume after creating it.

G. **invol** requests badspot information by asking whether or not you wish to use the prerecorded badspot list shipped with the disk. Answer **y**[es]. To erase the existing list, answer **no**. If you want to initialize the physical badspot list on a virgin disk, use option seven, not option one. Use option nine to add to an existing list. You must have a hardcopy of the badspots in order to enter them. **invol** has retained the badspot prompt in option one only for compatibility with existing shell files. After your affirmative response, **invol** displays the badspot list, indicating the physical disk address, cylinder, head, sector, and byte offset range.

If, in later operations, you wish to provide your own badspot information, these can be entered in one of several formats:

If the disk is a floppy: only HEX disk (block) addresses may be entered.

If a multi-disk set was assigned, only HEX physical disk (block) addresses can be entered (as reported by **salvol, lsyserr** or **disk_err**). These disk addresses are relative to the entire set. For example: for a multi-disk set, daddr 0 is on the 1st drive, 1 is on the 2nd one, etc.

Otherwise, the user has the option of entering the following:

HEX physical disk (block) addresses
DECIMAL cylinder-head byte_offset1 [byte_offset2 ...]

up to 8 byte offsets can be specified per-line (for the same cylinder/head)

HEX $cylinder-head sector1 [sector2 ...]

up to 8 sectors can be specified specified per-line (for the same cylinder/head)

Input continues until the user enters a blank line at which time he is given an opportunity to start over again (ignoring everything entered thus far).

If the disk contains logical volume badspots lists (see earlier) the badspot changes are propagated to these lists as well.

H. **invol** initializes the disk. As formatting proceeds, **invol** displays milestone messages to report its progress. It also displays a message for each volume it initializes, and another when it completes.

I. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

2 [–fnb5u]

Using option 2, you can partially initialize a volume, that is, add logical volumes to a physical volume, while preserving the existing logical volumes. Follow this procedure:

A. **invol** asks which option to perform. Type **2** to partially initialize a disk.

B. Specify the type of disk to initialize. (See option 1 step B.)

C. **invol** prints a list of the logical volumes and vacancies on the disk. If the disk has more than one vacancy, **invol** asks where to place the new logical volume by requesting a vacancy number. Indicate the vacancy that you want **invol** to use by entering its number. If there are logical volumes following the vacancy that you choose, **invol** prints a warning message and then automatically increments the volume numbers of those succeeding volumes by one.

D. Choose a verification option for the logical volume being initialized. (See option 1 step D.)

E. Enter the expected average file size, in blocks. (See option 1 step E.) Press <RETURN> for the default value, 5 blocks.

F. Enter the name and size of each logical volume to be formatted. (See option 1 step F.) After each specification, **invol** informs you of how much space is available. Terminate input with a blank line. A physical volume may have up to ten logical volumes.

G. Enter badspot information. (See option 1 step G.) Terminate badspot entry with a blank line.

H. Enter the name of the physical volume. (See option 1 step C.)

I. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

3 [–fnb5]

You can reinitialize a logical volume, retaining its size and name, with option 3.

All existing data in the volume will be lost. This option is useful for reinitializing floppy disks, where one logical volume typically occupies the entire physical volume.

To reinitialize a single logical volume, use the following procedure:

A. **invol** asks which option to perform. Type **3** to reinitialize a logical volume.

B. Specify the type of disk to initialize. (See option 1 step B.)

C. **invol** prompts for the # (1..N) of the logical volume to be re-initialized.

D. Choose a verification option: no verification, write all blocks, or write and reread all blocks. (See option 1 step D.)

E. Enter the expected average file size, in blocks. (See option 1 step E.) Press <RETURN> for the default value, 5 blocks.

F. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

4 Delete a logical volume.

To delete a logical volume, use the following procedure:

A. **invol** asks which option to perform. Type **4** to delete a logical volume.

B. Specify the type of disk from which the volume will be deleted. (See option 1 step B.)

C. Enter the number of the logical volume to delete. You can determine the logical volume numbers present on a disk with option 5.

D. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

Note: **invol** renumbers the logical volumes following the deleted volume.

5 Listing logical volumes

Logical volumes on the disk are displayed. Pre-SR10 format logical volumes are flagged as such in the output. To list the logical volumes on a disk, use the following procedure:

A. **invol** asks which option to perform. Type **5** to list the logical volumes on a disk.

B. Specify the type of disk. (See option 1 step B.) **invol** lists the volumes on that disk.

C. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

6 List badspots on disk or volume.

To list the badspots in one or more logical volumes, or for the physical volume, use the following procedure:

A. **invol** asks which option to perform. Type **6** to list badspots.

B. Specify the type of disk. (See option 1 step B.)

C. Specify the badspots to be listed, by entering one of the following:

m[**fg**] For ESDI drives only:
The contents of the manufacturer supplied badspot list is displayed. Physical disk (block) addresses are displayed in HEX along with the corresponding HEX (as opposed to decimal pre-SR10) cylinder/head/sector addresses and DECIMAL byte offset range.

Note: Users should generally have no reason to use this option as this list is usually copied to the physical badspot list/cylinder as soon as a disk is received. On some disks, moreover, this manufacturer's list is destroyed as soon as the disk is invol'd (option 1).

p[**hys**] For Winchester and Storage-Module drives only:
Displays the contents of the physical badspot list. Physical disk (block) addresses are displayed IN HEX along with the corresponding HEX (as opposed to decimal pre-SR10) cylinder/head/sector addresses and DECIMAL byte offset range.

If the specified disk is the primary drive for a multi-disk set, the physical disk addresses are relative to the entire set and a disk identifier is displayed along with the cylinder/head/sector. This disk identifier consists of a controller number and drive/unit number. For example:

```
phys  cylinder head sector byte offset C_num Drv_Unit
daddr                      range
...
...
123a9f 12d    e    5    7123-8034    0     2
```

If the specified disk is a non-primary member of a multi-disk set (i.e., the **s** option was used to examine a single drive only, see the **–m** option below), then the physical disk addresses that are

displayed are relative to that drive and do not correspond in any way to logical/physical disk addresses for the disk-set as a whole (i.e., as displayed by **lyserr** or **disk_err**).

n Badspots that lie within the specified logical volume are displayed: *n* can be any integer from 1 through 10. Both logical and physical disk addresses are displayed in HEX.

The specified disk must hold a valid pv and lv labels. The primary member of a multi-disk set must have been specified.

Note: cylinder/head/sector values are not displayed.

For a multi-disk set, the primary drive of the set must have been specified earlier. Physical disk addresses are relative to the entire set.

a[ll] Badspots for all logical volumes on the disk (or multi-disk set) are displayed: this format is identical with that of 1..10 discussed above.

D. **invol** asks if you have any more requests. Type **y[es]** to return to step A, or **n[o]** to return to the calling program (shell or Mnemonic Debugger).

7 Create physical badspot list.

Note: This option acts upon a single disk drive, regardless of whether it is a member of a multi-disk set or not. Generally speaking, this operation is run on a disk as it arrives from the factory and should not need to be performed again.
Using option 7, you can create or replace the badspot list on the disk. (Use option 9 to add badspots to an existing badspot list.)

A. **invol** asks which option to perform. Type **7** to enter the disk's badspot list.

B. Enter the location of the badspots, one per line. (See option 1 step G for the proper format.) Terminate badspot information with a blank line.

C. After you have typed in the list, **invol** asks you to check for errors. If you made any errors in the list, you must retype the entire list by returning to step A and beginning again.

D. **invol** asks if you have any more requests. Type **y[es]** to return to step A, or **n[o]** to return to the calling program (shell or Mnemonic Debugger).

8 Create or modify a Domain/OS paging file on an existing logical volume.

You can create an operating system file or modify the size of an existing one. The Domain/OS paging file is required if you intend to run the operating system

off of this logical volume.

To create or modify a Domain/OS paging file, perform the following steps:

A. **invol** asks which option to perform. Type **8**.

B. Specify disk type. (See option 1 step B.)

C. Specify logical volume number. The logical volumes present on a disk may be listed using option 5.

D. If a Domain/OS paging file already exists on this volume, **invol** displays the file's current size and asks if you want to change it. If you reply **y**[es], **invol** proceeds to step E. If you reply **n**[o], **invol** skips to step F.

E. **invol** prompts you to enter the number of pages you want in the OS paging file. Press <RETURN> to use the default, 352 pages. Type **0** (zero) to delete an existing paging file, or specify any number of pages between 1 and 288. If the size you enter is larger than the current Domain/OS paging file, **invol** displays milestones as it initializes new disk records.

F. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or type **n**[o] to return to the calling program (shell or Mnemonic Debugger).

9 Add to existing badspot list.

Using option 9, you can add to the disk's existing badspot list. Run **salvol** when this operation is done.

A. **invol** asks which option to perform. Type **9** to add to the disk's badspot list.

B. Enter the location of the badspots, one per line. (See option 1 step G for the proper format.) Terminate badspot information with a blank line.

C. After you have typed in the list, **invol** asks you to check for errors. If you made any errors in the list, you must retype the entire list by returning to step A and beginning again.

D. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

10 Display/change sector interleave factor for a logical volume.

Using option 10, you can set or display the sector interleave factor for a volume. The correct interleave factor is set when a logical volume is created. However, as performance improvements are made, it may become necessary to change it to achieve optimal block read/write rates. Operation 10 displays the current value and the optimal value which we recommend.

A. **invol** asks which option to perform. Type **10** to set or display the sector interleave factor.

B. Specify disk type. (See option 1 step B.)

C. **invol** displays a list of logical volumes for that physical volume. Specify the appropriate logical volume number. **invol** then displays the current sector interleave factor and the value which we recommend.

D. **invol** prompts for the new interleave factor. If you do not wish to change the interleave factor, enter a carriage return.

E. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or type **n**[o] to return to the calling program (shell or Mnemonic Debugger).

11 Remove badspots from existing badspot list.

Using option 11, you can subtract from the disk's existing badspot list. Run **salvol** when this operation is done.

A. **invol** asks which option to perform. Type **11** to remove from the disk's badspot list.

B. Enter the location of the badspots, one per line. (See option 1 step G for the proper format.) Terminate badspot information with a blank line.

C. After you have typed in the list, **invol** asks you to check for errors. If you made any errors in the list, you must retype the entire list by returning to step A and beginning again.

D. **invol** asks if you have any more requests. Type **y**[es] to return to step A, or **n**[o] to return to the calling program (shell or Mnemonic Debugger).

FULL DESCRIPTION OF FLAGS

–u Use defaults. You are prompted for as little information as possible. A physical volume is automatically chosen as follows (for option 1):

```
pv_wxy_year.month.day      w: disk type (or s or f)
                           x: controller #
                           y: unit #
                           year.month.day:  today's date
```

A single logical volume spanning the entire physical volume (for options 1 and 2) is constructed and chosen as follows:

```
lv_year.month.day
```

–n A non-bootable volume is constructed. By default, **invol** places the following objects on each logical volume that it constructs:

```
  lost+found.list       lost+found file for Salvol's later
                        use space is reserved for sysboot
* sysboot               (although the cpboot command must
                        be issued to actually install it
                        there)
  //                    local network root
  /                     entry directory for volume
* /sys
* /sys/node_data
* /sys/node_data/system_logs
```

If the **–n** flag is specified, the objects marked * are not placed on the disk. Although the disk can be mounted, it can not be used as a boot volume (unless it is re-invol'd).

–f Do not physically reformat the disk. By default, **invol** performs the very time-consuming task of re-formatting every track on the disk. This flag bypasses that operation and causes **invol** to execute very quickly, especially so if it is coupled with Verification option 1. Normally, it is only necessary to reformat disks as they arrive from the factory or after you have reason to suspect that the physical formatting is damaged or otherwise corrupted.

–b BSD style initial acls (for inheritance) are placed onto /sys and /sys/**node_data** when they are constructed:

```
owner/org:          ids from creating process
group:              id taken from parent directory
                    (set to wheel for /sys and
                    /sys/node_data)
<all> rights:       specified (usually from umask)
                    at create time
```

–5 SysV style initial acls (for inheritance) are placed onto /sys and /sys/**node_data** when they are constructed:

```
owner/group/org:    ids from creating process
                    rights specified (usually
                    from umask) at create time
world rights:         "          "          "
```

By default, Aegis inclusive initial acls are applied.

```
owner/group/org:    ids from creating process
                    rights set to [ignore]
world rights:       pwrx (wide-open)
```

–o Use only with option 1. A pre-SR10 format disk is configured which can be mounted under a pre-SR10 version of the operating system. By default, a SR10 format disk is created which cannot be mounted by a pre-SR10 node. The presence or absence of this flag controls whether subsequent logical volumes on this volume will have SR10 format (the new file system, acls and directories) or the pre-SR10 one.

–m Use after option 1 to group multiple physical disks together into a multi-disk set which thereafter will appear to be a single large disk (to **salvol, mtvol, dmtvol,** etc.).

If you answer the **invol** prompt with **1 –m** to indicate that a multi-disk group is to be configured, you will get the following prompts:

A. Select disk: [w=Winch|s=Storage mod|f=Floppy|q=Quit][ctrl#:][unit#]
invol prompts for the identity of the first of the disks. Assume that the user responds with **w0:1**. Subsequently, this drive becomes the primary disk of the set: it is the one which must be specified in order to mount or otherwise use the set.

B. How many disks will you be grouping together?
invol prompts for the number of disks to be collected into the set. Currently, the only legal responses are 1, 2 or 4.

C. Enter the striping option.
invol prompts for the algorithm to be used in spreading disk blocks across the multiple drives in the set:

1. Sector striping (subsequent sectors to different drives)
This is usually the choice for DN10000 workstations. Subsequent disk blocks are sent to alternate disk drives (e.g., disk block N is sent to a different drive from N-1 and N+1); this is the "classical" definition of disk striping.

2. Cylinder striping (subsequent cylinders to different drives)
This is usually the choice for 68000-based workstations. The cylinders from the various disks are "stacked" on top of one another so that it appears that each cylinder has N (2 or 4) times as many platters. Subsequent disk blocks will reside on the same drive unless a cylinder boundary is crossed, in which case we drop to the next disk in the set or wrap to the next cylinder of the first (primary) disk drive.

D. Physical volume name:
invol prompts for the name of the physical volume.
Enter remaining members of disk group:
Select disk: [w=Winch|s=Storage mod|f=Floppy|q=Quit][ctrl#:][unit#]
Select disk: [w=Winch|s=Storage mod|f=Floppy|q=Quit][ctrl#:][unit#]
invol prompts for the identity of the additional disk drives in the set. Obviously, no two drives in the set can have identical controller and unit numbers.
This **–m** flag for option 1 is allowable only if:

A. **invol** is running under an SR10 operating system.

B. An SR10 format disk is being configured.

C. A Winchester drive is selected. Only Winchester drives may be configured as a multi-disk set in this fashion. All drives in the set must have identical geometries: specifically, the blocks_per_track, sects_per_track, tracks_per_cylinder, blocks_per_volume, blocks_per_physical_volume, physical_badspot_daddr and physical_diagnostic_daddr attributes must all agree.

Note: Be aware that what is constructed is a single physical volume spanning multiple disk drives. Any logical volumes later built span all disk drives of the set. There is no way to cause a single file or logical volume to be limited to a single drive of the set. Contained within the pv label of the primary drive (as well as the others) is information identifying the other drives of the set as well as consistency data to detect the re-invol or replacement of any of the drives.

Relation of **invol** options to multi-disk sets:

The following **invol** options, when given the specifier of the primary drive of a set, will operate upon the entire set:

2 add a logical volume to an existing physical volume.
3 re-initialize an existing logical volume.
4 delete a logical volume.
5 list logical volumes.
6 list badspots: unless [s] is specified – <see below>.
8 create Domain/OS paging file.
9 add to badspot list: unless [s] is specified – <see below>.
10 changing interleave factor for a logical volume.
11 delete from badspot list: unless [s] is specified – <see below>.

Several **invol** options will only operate upon a single disk drive – regardless of

whether it is a member of a multi-disk group:

1 as detailed above, the specified drive is re-initialized regardless of its current status as a member of some multi-disk set ... optionally, a new multi-disk set can be established.
7 a physical badspot list on the specified drive is constructed (this option needs to be run when the drive arrives from the factory, and not again).

Several **invol** options allow the user to specify that only a single drive is to be assigned/processed, even if that drive is a member of a multi-disk set:

6 list badspots : only the physical (or manufacturer's) badspot list can be shown in this case.
9 add to badspot list : badspots may be entered as either physical disk addresses or as cylinder-head-sector triplets – both relative to that single drive as if it was not a member of a multi-disk set.
11 delete from badspot list : see the above discussion for 9.

For these cases, the "assign" prompt given to the user specifies that the user is allowed to follow the disk specifier with a " s" (<space> , "s") to indicate that even though the disk is a member of a multi-disk set, only the specified (single) disk is to be accessed. For example:

```
Select disk: [w=Winch|s=Storage mod|f=Floppy|q=Quit]
             [ctrl#:] [unit#]    w0:1 s
```

NAME

lb_admin – Location Broker Administrative Tool

SYNOPSIS

/etc/ncs/lb_admin

DESCRIPTION

The **lb_admin** tool monitors and administers Location Broker registrations. It presents both a Domain/Dialogue (tm) based user interface and a terminal-oriented interface. For information about the Domain/Dialogue interface, use the HELP key while running the tool. Information about individual commands for the terminal-oriented interface is available through the **help** command.

COMMANDS

help	List available commands or get information about a specific command.
quit	Exit the **lb_admin** session.
lookup	List matching Location Broker entries.
register	Add an entry to the Location Broker database.
set_broker	Select the specific Location Broker to use.
use_broker	Direct operations to a Local Location Broker or to a Global Location Broker.
unregister	Delete an entry from the Location Broker database.

NOTES

The Domain/Dialogue menus display object, type, and interface UUIDs when that operand is chosen via the switch at the top of the menu. The UUIDs displayed are those belonging to the entries of the broker selected in the top line when the update operation is performed. When an item is selected from a menu, the UUID appears in the Entry Information form and the next menu in the object, type, interface trio is displayed.

The asterisk (*) acts as a wildcard for the **lookup** and **unregister** operations.

SEE ALSO

drm_admin(8), glbd(8), llbd(8)

NAME

lcnet – display internet routing information

SYNOPSIS

/etc/lcnet [*options*]

DESCRIPTION

lcnet displays the list of known networks, their distance from the current node, the router used as the first hop from that network, and other information.

The distance (hops) from remote networks is measured in intervening routers. The distances are all for one-way traffic; if a network is three hops away from yours, your requests pass through three routers to get to that network. The responses also pass through three routers on the way back.

The **–conn** option shows you the full internet topology; that is, the list of networks and how the routers connect them together.

OPTIONS

–local (default)
Display the "First Hop" and "Hops" information for each network in the internet. The first hop is the node ID of a router on your network. It is the first router used in sending packets from your network to the target network. Other routers are also used if the target network is more than one hop away from your own.

–full Display information showing how up to date the routing table is (the "Age" and "Expiration" columns) in addition to the "First hop" and "Hops" information shown by the **–local** option. **–full** also lists inaccessible networks.

–conn Show which routers are connected to each network, and which other networks those routers touch. This option displays the "Touching" information.

–hw Display the type of hardware used for each of the networks (ring or IIC).

The **–conn** and **–hw** options may take several seconds to execute if you have a large internet.

–n *node-spec* Print another node's view of the internet. The outputs produced by **–local** and **–full** vary from node to node; **–n** affects these outputs. The **–n** option does not affect the output produced by the **–conn** or **–hw** options, since the hardware and connectivity do not depend on a node's position in the internet.

–c The **–c** option suppresses the title over each output column. It also fills every line of the "Network" column produced by the **–conn** option, and every line of the "Hardware" column produced by the **–hw** option. These format changes make it easier to use **lcnet**'s output as another program's input.

EXAMPLES

In this example, the node is directly connected to network C0FFEE. Networks 5A1AD and ED1F1CE were connected in the past, but are not now accessible (perhaps because the routers are down).

The expiration date and time for the "local" network are meaningless.

```
$ /etc/lcnet –full
            First
Network      Hop      Hops     Age  Expiration date/time
========    =====    =====     ===  ====================
    B020     4B6F        1     NEW  1987/06/16 14:33:21
  B00B00     4B6F        2     NEW  1987/06/16 14:33:21
   5A1AD     4B6F     gone     NEW  1987/06/16 14:33:21
  C0FFEE        0    local     NEW  1987/06/09 10:27:46
 ED1F1CE     4B6F     gone     NEW  1987/06/16 14:33:21
    D0D0     BAD1        1     NEW  1987/06/16 14:33:39
```

The "Touching" information describes your internet completely. This example includes several kinds of information:

```
- Network DEFACED has one router, node 2A3B.
  That router connects DEFACED to EFFACED.
- Network FACE0FF contains two routers, 31DC and 1371.
  Those routers connect FACE0FF to C0C0A and C0FFEE,
  respectively.
```

$ /etc/lcnet –conn

```
                Touching      Touching
Network          Router       Network
========      ========      ========
    F00D          5C0B          DECAF
                  36CF         C0FFEE
   5A1AD          459B         C0FFEE
                  45BE        ED1F1CE
   B002E          3F0A         C0FFEE
   C0C0A          BAD1         B00B1E
                  56B0        EFFACED
                  31DC        FACE0FF
   DECAF          5C0B            F00D
  B00B1E          BAD1           C0C0A
  C0FFEE          36CF            F00D
                  459B           5A1AD
                  3F0A           B002E
                  1371        FACE0FF
 DEFACED          2A3B        EFFACED
 ED1F1CE          45BE           5A1AD
 EFFACED          56B0           C0C0A
                  2A3B        DEFACED
 FACE0FF          31DC           C0C0A
                  1371         C0FFEE
```

$ /etc/lcnet –conn –c

```
    F00D          5C0B          DECAF
    F00D          36CF         C0FFEE
   5A1AD          459B         C0FFEE
   5A1AD          45BE        ED1F1CE
   B002E          3F0A         C0FFEE
   C0C0A          BAD1         B00B1E
   C0C0A          56B0        EFFACED
   C0C0A          31DC        FACE0FF
   DECAF          5C0B            F00D
  B00B1E          BAD1           C0C0A
  C0FFEE          36CF            F00D
  C0FFEE          459B           5A1AD
  C0FFEE          3F0A           B002E
  C0FFEE          1371        FACE0FF
 DEFACED          2A3B        EFFACED
 ED1F1CE          45BE           5A1AD
 EFFACED          56B0           C0C0A
```

```
EFFACED         2A3B      DEFACED
FACE0FF         31DC        C0C0A
FACEOFF         1371       C0FFEE
```

NAME

lcnode – list nodes connected to the network

SYNOPSIS

/etc/lcnode [*options*]

DESCRIPTION

lcnode lists the nodes currently connected to the network. The list contains the ID of every node connected, the time at which the node was started, the current time, and the name of each node's entry directory.

This command reports only the nodes that respond within a preset time limit. If a node is connected, but temporarily unable to respond within the specified time, it does not appear in the produced list.

OPTIONS

–m[e] Request information about your node only. This option displays the node ID.

–b[rief] Request brief output. **lcnode** lists only the entry directory name for each connected node. Note that the entry directory of a diskless node is the entry directory of its paging partner.

–id When used with **–brief**, display the node ID in addition to the entry directory.

–c[ount] Request node count only. **lcnode** lists only the number of nodes responding to its query.

–max[nodes] *n* Set a limit on the number of nodes you want to see, even if more could respond.

–from *node_spec* Starts the node list at some node other than your own. This is especially useful in an internet environment, for looking at networks other than your own. See **help node_spec** for details about node specification syntax.

–name When you specify the **–brief** option, **lcnode** normally prints the entry directory for each node. If you specify **–name** with **–brief**, **lcnode** prints the node name cataloged with the naming server. Only diskless nodes are printed differently. A diskless node's entry directory is its partner's node name; a diskless node's node name is uniquely its own.

Unless the **–from** option specifies your own node, the list includes only an unbroken sequence of nodes running Aegis SR9.0 or later. The rest of the node list is lost, starting with the first node running a pre-SR9.0 Aegis.

EXAMPLES

1. $ **/etc/lcnode**

```
The node ID of this node is 21.  3 other nodes responded.

id        Boot time        Current time      Entry Directory

21  1987/06/09  9:21:44 1987/06/09 16:06:22  //dollar
17  1987/06/09 13:52:02 1987/06/09 16:06:13  //quarter
27  1987/06/09 12:53:28 1987/06/09 16:06:07  //nickel
11  1987/06/09 12:03:39 1987/06/09 16:06:15  ** DISKLESS **
                           //diskless_$11 partner node: 17
```

2. $ **/etc/lcnode –me**

```
The node id of this node is 21.
```

3. $ **/etc/lcnode –b**

```
//dollar
//quarter
//nickel
//quarter
```

(//quarter appears once as the host for a diskless node and once for the node with the disk.)

4. $ **/etc/lcnode –b –name**

```
//dollar
//quarter
//nickel
//diskless_$000011
```

(–name shows you the name under which diskless node 11 is cataloged)

5. $ **/etc/lcnode –c**

```
466 other nodes responded.
```

6. $ **/etc/lcnode –c –b**

```
466
```

7. **$ /etc/lcnode –c –m**

```
The node id of this node is 116A.
466 other nodes responded.
```

8. **$ /etc/lcnode –b –id**

```
21  //dollar
17  //quarter
27  //nickel
11  //quarter
```

9. **$ /etc/lcnode –from 0FAD.3924 –max 2**

```
Starting from node 3924.
1 other node responded,
   but more might have responded with a high -max value.

Node id  Boot time             Current time      Entry Directory

 3924   1985/02/14 17:20:45   1985/02/14 19:07:04  //laurel
 34Bf   1985/02/14 18:46:52   1985/02/14 19:08:09  //hardy
```

NAME

llbd – Local Location Broker Daemon

SYNOPSIS

/etc/ncs/llbd

DESCRIPTION

The Local Location Broker Daemon (**llbd**) is part of the Network Computing System (NCS). It manages the Local Location Broker (LLB) database, which stores information about NCS-based server programs running on the local node.

A host must run **llbd** if it is to support the Location Broker forwarding function or to allow remote access (e.g., by the **lb_admin** tool) to the LLB database. In general, any node that runs an NCS-based server program should run an **llbd**. Additionally, any network supporting NCS activity should have at least one node running the Global Location Broker Daemon (**glbd**). Typically, both daemons are started at boot time from the /etc/rc file.

To start **llbd** on a node that is already running, type the following at a shell prompt:

```
$ /etc/server /etc/ncs/llbd &
```

To have **llbd** start every time a node boots, use **touch** or **crf** to create the file /etc/daemons/llbd.

If **llbd** is to support remote access from hosts in the IP address family, a TCP daemon (**tcpd**) must be running on the local node; **tcpd** should be started before **llbd**.

NAME

login_sh – start a log-in shell

SYNOPSIS

/sys/dm/login_sh

DESCRIPTION

The **login_sh** command is included in your Display Manager (DM) log-in start-up script to create a shell when you log in.

When you log in to the DM, it:

- Checks the **/etc/passwd** file (the registry) to determine your HOME directory and preferred SHELL.
- Looks for your system type by reading the **.systype** file in your home directory. If you don't have a **.systype** file, **login_sh** reads the file **/etc/environ** and uses the systype listed in that file. Valid systypes are **bsd4.3**, **sys5.3**, and **aegis**.
- Prints the "message of the day" from the **/etc/motd** file and checks for UNIX mail if your login shell is **csh** or **sh** (**/etc/profile** prints the message of the day for **ksh**).
- Executes the specified shell as a log-in shell. If the Bourne or the Korn shells (**/bin/sh** or **/bin/ksh**) are run as log-in shells, they execute **/etc/profile** and **~/.profile**. The C shell (**/bin/csh**) executes the **~/.login** file.

If **/etc/passwd** doesn't specify a shell, **login_sh** runs **sh**.

NOTES

login_sh is intended to replace the **start_sh** and **start_csh** commands. To avoid executing the log-in scripts **/etc/profile**, **.profile**, or **.login** more than once, you should create only one log-in shell.

The **/etc/environ** file establishes the default environment and default systype for the node. The **/etc/environ** file can contain two lines, specifying the ENVIRONMENT value, and the SYSTYPE values. Valid ENVIRONMENT values are: **aegis**, **bsd**, or **sysv**. Valid SYSTYPE values are: **aegis**, **bsd4.3**, or **sys5.3**. If the ENVIRONMENT is set to **aegis**, you can add a line specifying either **bsd4.3** or **sys5.3**. For example, if the file contains the following lines, **login_sh** executes an Aegis shell (unless another shell is specified in the registry) and sets the systype to **sys5.3**.

ENVIRONMENT = aegis
SYSTYPE = sys5.3

FILES

/bin/csh
/bin/ksh
/bin/sh
/etc/motd
/etc/profile
/etc/environ
~/.systype

SEE ALSO

environment(1), csh(1), sh(1), ksh(1)

NAME

lpc – line printer control program

SYNOPSIS

/etc/lpc [*command* [*argument* ...]]

DESCRIPTION

The **lpc** command is used by the system administrator to control the operation of the line printer system. For each line printer configured in **/etc/printcap**, **lpc** may be used to:

- Disable or enable a printer
- Disable or enable a printer's spooling queue
- Rearrange the order of jobs in a spooling queue
- Find the status of printers, and their associated spooling queues and printer dameons.

Without any arguments, **lpc** will prompt for commands from the standard input. If arguments are supplied, **lpc** interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected, causing **lpc** to read commands from file.

COMMANDS

Commands may be abbreviated; the following is the list of recognized commands.

? [*command ...*]

help [*command ...*]

Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort { *all* | *printer ...* }

Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by **lpr**) for the specified printers.

clean { *all* | *printer ...* }

Remove any temporary files, data files, and control files that cannot be printed (i.e., do not form a complete printer job) from the specified printer queue(s) on the local machine.

disable { *all* | *printer ...* }

Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by **lpr**.

down { *all* | *printer* } *message ...*

Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like **echo**(1). This is normally used to take a printer down

and let others know why (**lpq** will indicate the printer is down and print the status message).

enable { *all* | *printer* ... }
Enable spooling on the local queue for the listed printers. This will allow **lpr** to put new jobs in the spool queue.

exit

quit
Exit from **lpc.**

restart { *all* | *printer* ... }
Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. **lpq** will report that there is no daemon present when this condition occurs. If the user is the super-user, try to abort the current daemon first (i.e., kill and restart a stuck daemon).

start { *all* | *printer* ... }
Enable printing and start a spooling daemon for the listed printers.

status { *all* | *printer* ... }
Display the status of daemons and queues on the local machine.

stop { *all* | *printer* ... }
Stop a spooling daemon after the current job completes and disable printing.

topq *printer* [*jobnum* ...] [*user* ...]
Place the jobs in the order listed at the top of the printer queue.

up { *all* | *printer* ... }
Enable everything and start a new printer daemon. Undoes the effects of **down**.

FILES

/etc/printcap	printer description file
/usr/spool/lpd	spool directories
/usr/spool/lpd/lock	lock file for queue control

DIAGNOSTICS

''?Ambiguous command''	abbreviation matches more than one command
''?Invalid command''	no match was found
''?Privileged command''	command can be executed by root only

SEE ALSO

lpd(8), lpr(1), lpq(1), lprm(1), printcap(5)

NAME

lpd – line printer daemon

SYNOPSIS

/usr/lib/lpd [**–l**] [*port#*]

DESCRIPTION

lpd is the line printer daemon (spool area handler). It is normally invoked at boot time from the **rc**(8) file. It makes a single pass through the **printcap**(5) file to find out about the existing printers, and prints any files left after a crash. It then uses the system calls **listen**(2) and **accept**(2) to receive requests to print files in the queue, transfer files to the spooling area, display the queue, or remove jobs from the queue. In each case, it forks a child to handle the request so the parent can continue to listen for more requests. The Internet port number used to rendezvous with other processes is normally obtained with **getservbyname**(3) but can be changed with the *port#* argument.

Access control is provided by two means. First, all requests must come from one of the machines listed in the file **/etc/hosts.equiv** or **/etc/hosts.lpd**. Second, if the ''rs'' capability is specified in the **printcap** entry for the printer being accessed, **lpr** requests will only be honored for those users with accounts on the machine with the printer.

The file named **minfree** in each spool directory contains the number of disk blocks to leave free so that the line printer queue won't completely fill the disk. The **minfree** file can be edited with your favorite text editor.

The file named **lock** in each spool directory is used to prevent multiple daemons from becoming active simultaneously, and to store information about the daemon process for **lpr**(1), **lpq**(1), and **lprm**(1). After the daemon has successfully set the lock, it scans the directory for files beginning with **cf**. Lines in each **cf** file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character to specify what to do with the remainder of the line (see ''KEY CHARACTERS'', below).

If **lpd** can't open a file, it logs a message via **syslog**(3) using the **LOG_LPR** facility. **lpd** will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.

lpd uses **flock**(2) to provide exclusive access to the lock file and to prevent multiple deamons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control filename of the current job being printed. The second line is updated to reflect the current status of **lpd** for the programs **lpq**(1) and **lprm**(1).

OPTIONS

port#	Change the Internet port number used to rendezvous with other processes to *port#*.
–l	Log valid requests received from the network. This option can be useful for debugging purposes.

KEY CHARACTERS

J	Job Name. A string to be used for the job name on the burst page.
C	Classification. A string to be used for the classification line on the burst page.
L	Literal. The line contains identification information from the password file, and causes the banner page to be printed.
T	Title. A string to be used as the title for **pr**(1).
H	Host Name. The name of the machine where **lpr** was invoked.
P	Person. The login name of the person who invoked **lpr**. This is used to verify ownership by **lprm**.
M	Send mail to the specified user when the current print job completes.
f	Formatted File. The name of a file to print which is already formatted.
l	Like **f**, but passes control characters and does not make page breaks.
p	The name of a file to print using **pr**(1) as a filter.
t	**troff** File. The file contains **troff**(1) output (phototypesetter commands).
n	Ditroff File. The file contains device independent **troff** output.
d	DVI File. The file contains **Tex**(1) output (DVI format from Standford).
g	Graph File. The file contains data produced by **plot**(3X).
c	Cifplot File. The file contains data produced by **cifplot**.
v	The file contains a raster image.
r	The file contains text data with FORTRAN carriage control characters.
1	**troff** Font R. The name of the font file to use instead of the default.
2	**troff** Font I. The name of the font file to use instead of the default.
3	**troff** Font B. The name of the font file to use instead of the default.
4	**troff** Font S. The name of the font file to use instead of the default.
W	Width. Changes the page width (in characters) used by **pr**(1) and the text filters.
I	Indent. The number of characters to indent the output by (in ascii).

U Unlink. The name of a file to remove upon completion of printing.

N Filename. The name of the file which is being printed, or a blank for the standard input (when **lpr** is invoked in a pipeline).

FILES

/etc/printcap	printer description file
/usr/spool/lpd	spool directories
/usr/spool/lpd/minfree	minimum free space to leave
/usr/spool/lpd/*name*	hostname of machine to run **lpd**
/dev/lp*	line printer devices
/dev/printer	socket for local requests
/etc/hosts.equiv	lists machine names allowed printer access
/etc/hosts.lpd	lists machine names allowed printer access, but not under same administrative control.

SEE ALSO

llbd(8), lpc(8), lpr(1), lpq(1), lprm(1), syslog(3), printcap(5)

NAME

lprotect – control local protection

SYNOPSIS

/etc/lprotect [–e rootlocal] [–d rootlocal]

DESCRIPTION

The **lprotect** command controls local protection attributes on a node. Currently, this command enables requests by root (locksmith) to be honored only if they originate locally (**rootlocal**), i.e. from a local process. If no options are specified, the current state of **rootlocal** is returned. To change the state of the **rootlocal** attribute, you must be running as root (locksmith).

OPTIONS

–e rootlocal	Enables local-only root requests.
–d rootlocal	Disables local-only root requests.

EXAMPLE

1. Show current status.

```
$ /etc/lprotect
"local-only root requests" is disabled. (-d rootlocal)
```

2. Enable local root requests

```
$ /etc/lprotect –e rootlocal
$ /etc/lprotect
 "local-only root requests" is enabled.
```

3. Disable local root requests

```
$ /etc/lprotect –d rootlocal
```

NAME

makedev – make system special files

SYNOPSIS

/dev/MAKEDEV *device...*

DESCRIPTION

MAKEDEV is a shell script normally used to install special files. It resides in the **/dev** directory, as this is the normal location of special files.

ARGUMENTS

Arguments to **MAKEDEV** are usually of the form *device-name?* where *device-name* is one of the supported devices and *?* is a logical unit number (0-9). A few special arguments create assorted collections of devices and are listed below.

std Create the *standard* devices for the system; e.g. **/dev/console**, **/dev/tty**.

local Create those devices specific to the local site. This request causes the shell file **/dev/MAKEDEV.local** to be executed. Site specific commands, such as those used to setup dialup lines as **ttyd***?* should be included in this file.

NOTES

Since all devices are created using **mknod**(8), this shell script is useful only to the super-user.

DIAGNOSTICS

Either self-explanatory, or generated by one of the programs called from the script. Use **sh -x MAKEDEV** in case of trouble.

SEE ALSO

intro(4), mknod(8)

NAME

mbd – dump usage info on tcp buffer pool

SYNOPSIS

/etc/mbd [**–f**] [**–k**]

DESCRIPTION

The **mbd** command dumps usage information about the tcp memory buffer pools. Usage statistics on tcp memory buffers may be obtained by using the **-m** option of the **netstat** command; **mbd** is intended for analyzing cases where buffers are being lost. It scans the entire buffer pool, finding all the chains of in-use buffers; it then prints each chain of buffers. This information may help you in discovering reasons why buffers are being lost.

OPTIONS

–f Dump the free pools as well as the chains of in-use buffers. This produces a lot of output.

–k Don't try to lock the mutex on the buffer pools before doing the dump. This is useful mostly when the **tcpd** has crashed with the buffer pool mutex locked.

EXAMPLES

A dump of the buffer pools of a basically idle tcp might look like this:

```
$ /etc/mbd
Offset 0x000035cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x000041cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00003bcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x0000a7cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00004dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00007dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
```

Here there are 6 large (1520-byte) buffers in use, all on a single chain.

NAME

mkcon – set console device

SYNOPSIS

/etc/mkcon [**–p**] [**–d** *dev*] [**–c** *cmd*] [**–n**]

DESCRIPTION

If no arguments are specified with this command, it makes the current controlling terminal into a console and starts up a shell. The shell type is determined by the shell environment variable. When the shell exits, the console output is redirected to *'node_data/system_logs/console*.

OPTIONS

–p	Create a new DM pad for the console in place of the controlling terminal device.
–d *dev*	Make *dev* replace the controlling terminal device as console.
–c *cmd*	Execute *cmd* instead of $SHELL.
–n	Do not run a shell.

EXAMPLE

$**/etc/mkcon –d /dev/display –n**

This causes console output to appear in a DM window, with a new window each time */dev/console* is opened.

NAME

mkdev – shell script to make devices

SYNOPSIS

mkdev *device_directory* [**–d** *devno_file*] [*all* / *console* / *tty* / *null* / *sio* / *pad* / *pty* / *dsk* / *mt* / *global_devices* / *crp*]

DESCRIPTION

mkdev creates devices. *device_directory* is usually /**dev**; **–d** *devno_file* can be used to specify a device number/manager mapping file to use in place of *'node_data/device_numbers.*

If no additional arguments are specified, then all devices which have not been created will be. **mkdev** creates a file called **.mkdev** in the device directory to record for future instances of itself what work has to be done, (actually just the version of the last **mkdev** to run to completion).

If any arguments are specied (or *all*) then these devices will be deleted and recreated.

DIAGNOSTICS

Should be self-explanatory.

FILES

'node_data/device_numbers Default device number mapping file

SEE ALSO

mkdevno(1M)

NAME

mkhosts – generate hashed host table

SYNOPSIS

/etc/mkhosts [**–v**] [*hostfile*]

DESCRIPTION

The **mkhosts** command is used to generate the hashed host database used by one version of the library routines **gethostbyaddr**(3N) and **gethostbyname**(3N). It is not used if host name translation is performed by **named**(8).

The new database is build in a set of temporary files and only replaces the real database if the new one is built without errors. **mkhosts** will exit with a non-zero exit code if any errors are detected.

OPTIONS

–v List each host as it is added. The default *hostfile* is **/etc/hosts**. If another file is specified, it must be in the format of **/etc/hosts** (see **hosts**(5)). **mkhosts** will generate database files named *hostfile*.**pag** and *hostfile*.**dir**.

FILES

hostfile.**pag** - real database filenames
hostfile.**dir**
hostfile.**new.pag** - temporary database filenames
hostfile.**new.dir**

SEE ALSO

gethostbyname(3), gettable(8), hosts(5), htable(8), named(8);
Configuring and Managing TCP/IP.

NAME

mknod – build special file

SYNOPSIS

/etc/mknod *name* [**c**] [**b**] *major minor*

DESCRIPTION

mknod makes a special file. The first argument is the *name* of the entry. The second is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g. unit, drive, or line number).

The assignment of major device numbers is specific to each system. They have to be dug out of the system source file **conf.c**.

mknodx is a variation of **mknod**. This command is used by Apollo system administrators, and should not be deleted.

SEE ALSO

mknod(2), makedev(8)

NAME

mount, umount – mount and dismount file system

SYNOPSIS

/etc/mount [*special name* [**–r**]]

/etc/mount –a

/etc/umount *special*

/etc/umount –a

DESCRIPTION

mount announces to the system that a removable file system is present on the device *special*. The file *name* must exist already; it must be a directory (unless the root of the mounted file system is not a directory). It becomes the name of the newly mounted root.

umount announces to the system that the removable file system previously mounted on device *special* is to be removed.

These commands maintain a table of mounted devices in **/etc/mtab.** If invoked without an argument, **mount** prints the table.

Physically write-protected and magnetic tape file systems must be mounted read-only or errors will occur when access times are updated, whether or not any explicit write is attempted.

OPTIONS

–r Indicates that the file system is to be mounted read-only.

–a If this option is present for either **mount** or **umount**, they attempt to mount or unmount all of the file systems described in **/etc/fstab**. In this case, *special* and *name* are taken from **/etc/fstab.** The *special* file name from **/etc/fstab** is the block special name.

FILES

/etc/mtab	mount table
/etc/fstab	file system table

CAUTIONS

Mounting file systems full of garbage will crash the system.
Mounting a root directory on a non-directory makes some apparently good pathnames invalid.

SEE ALSO

mount(2), mtab(5), fstab(5)

NAME

named – internet domain name server

SYNOPSIS

named [**–d** *debuglevel*] [**–p** *port#*] [*bootfile*]

DESCRIPTION

named is the internet domain name server (see RFC883 for more details). Without any arguments, **named** reads the default boot file **/etc/named.boot**, reads any initial data, and listens for queries.

OPTIONS

–d *debuglevel* Print debugging information. A number after the **d** determines the level of messages printed.

–p *port#* Use a different port number. The default is the standard port number as listed in **/etc/services**.

bootfile Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data.

EXAMPLE

The following example shows a boot file:

```
;
;      boot file for name server
;
; type        domain        source file or host
;
domain        berkeley.edu
primary             berkeley.edu     named.db
secondary     cc.berkeley.edu 10.2.0.78 128.32.0.10
cache                          named.ca
```

The first line specifies that **berkeley.edu** is the domain for which the server is authoritative. The second line states that the file **named.db** contains authoritative data for the domain **berkeley.edu**. The file **named.db** contains data in the master file format described in RFC883, except that all domain names are relative to the origin; in this case, **berkeley.edu** (see below for a more detailed description).

The third line specifies that all authoritative data under **cc.berkeley.edu** is to be transferred from the name server at 10.2.0.78. If the transfer fails it will try 128.32.0.10 and continue trying the address, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain.

The fourth line specifies data in **named.ca** is to be placed in the cache (i.e., well known data such as locations of root domain servers). The file **named.ca** is in the same format as **named.db**.

MASTER FILE FORMAT

The master file consists of entries of the form:

$INCLUDE *<filename>*
$ORIGIN *<domain>*
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>

where *domain* is dot " ." for root, "@" for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with ".", the current origin is appended to the domain. Domain names ending with "." are unmodified. The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero. The *opt_class* field is the object address type; currently only one type is supported, **IN**, for objects connected to the DARPA internet. The *type* field is one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A	A host address (dotted quad)
NS	An authoritative name server (domain)
MX	A mail exchanger (domain)
CNAME	The canonical name for an alias (domain)
SOA	Marks the start of a zone of authority (5 numbers (see RFC883))
MB	A mailbox domain name (domain)
MG	A mail group member (domain)
MR	A mail rename domain name (domain)
NULL	A null resource record (no format or data)
WKS	A well-known service description (not implemented yet)
PTR	A domain name pointer (domain)
HINFO	Host information (cpu_type OS_type)
MINFO	Mailbox or mail list information (request_domain error_domain)

NOTES

The following signals have the specified effect when sent to the server process using the **kill**(1) command.

SIGHUP Causes server to read **named.boot** and reload database.

SIGINT Dumps current data base and cache to **/usr/tmp/named_dump.db**

SIGUSR1 Turns on debugging; each SIGUSR1 increments debug level.

SIGUSR2 Turns off debugging completely.

FILES

/etc/named.boot	name server configuration boot file
/etc/named.conf	configuration file
/etc/named.pid	the process id
/usr/tmp/named.run	debug output
/usr/tmp/named_dump.db	dump of the name servers database

Configuration files read by **/etc/named.boot**:
/etc/named.ca
/etc/named.hosts
/etc/named.local
/etc/named.rev

SEE ALSO

kill(1), gethostbyname(3N), signal(3c), resolver(3), resolver(5);
Configuring and Managing TCP/IP;
RFC882, RFC883, RFC973, RFC974, *Name Server Operations Guide for BIND*.

NAME

netmain – analyze network maintenance stats

SYNOPSIS

/etc/netmain [*options*]

DESCRIPTION

netmain is an interactive, menu-driven program that lets you control the **netmain_srvr**, the network maintenance server, and analyze the data that **netmain_srvr** produces. **netmain** provides detailed help from its menus.

OPTIONS

–w[help] (default) Make sure the window is large enough to display command menus and interactive help.

–wc[md] Set the window size smaller for command menus only. If you later decide that you want to see the helps, grow the window manually with <GROW>.

–nw Do not change the window size.

EXAMPLES

1. Run **netmain** in a window large enough to display command menus and interactive help:

   $ **/etc/netmain**

2. Run **netmain** in a window large enough (but no larger) to display the command menus:

   $ **/etc/netmain –wc**

NAME

netmain_chklog – clean up bad log files

SYNOPSIS

/etc/netmain_chklog *[options]* pathname...

DESCRIPTION

When the **netmain_srvr** program halts catastrophically (for instance, during a node reset), it can leave the log file it was writing in a corrupt, unusable state. **netmain_chklog** determines whether the log is corrupt and, optionally, deletes corrupt files.

If the pathname you specify points to some kind of file other than a **netmain** log file, that file is almost always ignored; it is almost never deleted as a corrupt log. On very rare occasions, another kind of file may look so much like a corrupt log that it might be deleted accidentally if you use both **–d** and the standard command option **–nq** (no query). Thus you should use **–d –nq** with extreme care.

pathname (required) Specify the files to be checked. Multiple names and wildcarding are permitted; separate names with blanks.

OPTIONS

–d	Delete corrupt log files.
–nd (default)	Do not delete anything.
–l (default)	Describe every file analyzed.
–nl	Describe only corrupt log files.

EXAMPLES

$ **/etc/netmain_chklog 'node_data/net_log/***

NAME

netmain_note – place message in network error log

SYNOPSIS

/etc/netmain_note *string* [*string* ...]

DESCRIPTION

netmain_note sends a text string to **netmain_srvr**, the network maintenance server. The message is broadcast to all maintenance servers.

Typical topics of maintenance notes include known or explainable network failures, scheduled down-time, and node installations.

string (required) Specify message to be sent. You may use any string that is legal in a shell command. (Note that the shell takes special action on some keywords, such as 'if', unless you place them in quotation marks.) If there is more than one string, **netmain** builds the note by concatenating the arguments that are separated by spaces.

EXAMPLES

```
$ /etc/netmain_note 'Scheduled down time at 5 pm.'

$ /etc/netmain_note Cable disconnected at //sancho_panza
```

NAME

netmain_srvr – collect network error stats

SYNOPSIS

/etc/netmain_srvr [options] [*pathname*]

DESCRIPTION

netmain_srvr collects and stores performance statistics for the Apollo token ring network. Use **netmain_srvr** to gather information; use the **netmain** program to display and analyze the information.

You can set parameters for **netmain_srvr** from the command line and from an options file. Once the server is running, you can change any parameter using the **netmain** program. To include parameters in an options file, specify the **–cmdf** option.

When you specify **–cmdf** *pathname*, **netmain_srvr** reads the options listed in the options file first and then reads any other options on the **netmain_srvr** command line. If options specified in the file and on the command line differ, **netmain_srvr** uses the command line settings. For example, if the options file specifies a log file length as **–ll 1500**, and the command line specifies **–ll 3000**, **netmain_srvr** uses **–ll 3000**.

If a **netmain_srvr** does not start properly, a record of the failure appears in **'node_data/netmain_srvr.err_log**.

OPTIONS

–a[ppend] Append to an existing log file with this name, if one already exists; otherwise, create a log file with this name. This option is only valid when a log file pathname is specified with the **–l** option. Contrast this with the **–nappend** option.

–cmdf *pathname* Accept options from an ASCII text file *pathname*. You may use this option only from the command line, not in the options file. There can only be one options file.

–l[og] *[pathname]* (default) Create a log file. Optionally, specify a pathname, which is relative to the **'node_data/net_log** directory. If either this option or the pathname is not specified, the log file name is derived from the current date: **'node_data/net_log/net_log.yy.mm.dd.** The log file is stored on the disk of the node running **netmain_srvr**, and must remain there for **netmain_srvr** to write to it.

–ll *n* (default) Set an upper limit on the length of the log file. The file size limit *n* is in 1024-byte units. The default value for *n* is 3000. You must use this option when you start the monitor and if you don't want to use the default length for the first log file, since you cannot change the name of a log file once it's open.

–na[ppend] (default)
Create a new log file, over-writing any log with that name, if one exists. This option is only valid when a log file pathname is specified with the **–l** option. Contrast this with the **–append** option.

–nl[log] Do not write to a log file. **netmain_srvr** still runs probes and observers.

–ntopo[_init] (default)
Override the **–topo_init** option, if **–topo_init** is specified in an options file. **–ntopo** is useful only on the command line.

–obs[erve] *observer time ...*
Set the interval at which the named observer wakes up. Specify *time* as hh:mm:ss, hh:mm, or *never*, if you do not want the monitor to run the observer. Multiple observer/time pairs are permitted. See the default times listed below for each observer.

–re_obs[erve] *observer time ...*
Set the "Recheck interval" — the interval that the observer waits before rechecking a node that has caused an alarm condition. By setting a recheck interval, you ensure that the observer only reports on a node once every *time* period. If the recheck interval is too short, the observer may produce many redundant alarms. Specify *time* as hh:mm:ss, or hh:mm. Multiple observer/time pairs are permitted. See the default recheck intervals listed below for each observer.

–s[ample] *probe time ...*
Set the interval at which the named probe wakes up. Specify *time* as hh:mm:ss, hh:mm, or **never**, if you do not want the monitor to run the probe. Multiple probe/time pairs are permitted. See the default times listed below for each probe.

–sk[ip] *probe distance ...*
Set the skip distance for the probe named. If the skip distance is *n*, the named probe samples approximately 1/*n* of the nodes every time it wakes up. Multiple probe/distance pairs are permitted. See the default skip distances listed below for each probe.

–topo[_init] *pathname*
Initialize the monitor's total node list from a data file. The file may contain any number of node names or hexadecimal IDs, separated by spaces or on separate lines. If there is a "#" or "{" in any line, that character and all characters to the right of it are ignored (that is, "#" and "{" are comment markers).

NAME

netsvc – set or display network services

SYNOPSIS

/etc/netsvc [*options*]

DESCRIPTION

netsvc sets or displays the network services that this node will perform. All changes take place immediately.

OPTIONS

If no options are specified, **netsvc** displays the network services allowed for this node.

–n None. Disable all network services and physically disconnect this node from the network.

–l Local. Allow only network requests originating at this node.

–r Remote. Allow only network requests originating at other nodes.

–a (default) All. Allow both locally and remotely initiated network requests. (The size of the remote paging pool is not changed.)

–s[ervers] [*n*] Servers. Set the number of network servers to run on this node. At system startup, the number of network servers is 1. If this node is a network partner for diskless nodes or has several remote file users, their performance can be improved by increasing the number of servers. If *n* is not specified, the maximum number of servers (3) is used.

–p [*n*] Pool. Set local memory pool size. Network page requests originating at remote nodes may not use more than *n* pages of the local node's memory. If *n* is not specified, all the local node's memory is eligible for remote page requests.

–net [*net_id*] Network ID. Set or display network ID. Use this option to change or examine the ID of the network to which the node is attached. It affects only the node at which you type the command, not the rest of the network. Specifying a hexadecimal network ID changes your node's network ID. Using **–net** with no argument forces **netsvc** to display your network ID even if it is set to 0.

This option is useful only when there are no internet routers active on the node's network. Routers give the network ID to nonrouting nodes every 30 seconds, and may override the network ID you specify with this option.

NOTE

If the network ID you set with **–net** differs from the network ID used by other nodes on your network, your node may not be able to communicate with those other nodes.

Be careful when revoking network access with **–n** or **–l**. Remote file users may have problems, and writable files may be damaged. If your node was the network partner for a diskless node, that node will crash when your node leaves the network.

Use the –s option carefully. Although you can increase the number of servers, you cannot decrease it. The only way to return to a smaller number of servers is to reboot the node. Also note that increasing the number of server processes decreases the number of user processes allowed.

EXAMPLES

```
$ /etc/netsvc
Network operations allowed: ALL
Number of network servers: 1
Remotely initiated paging pool limit: NONE
Network ID: 437A9
$
```

SEE ALSO

More information is available. Type

help rtsvc For information about controlling a node's internet routing service

NAME

nodestat – display network statistics

SYNOPSIS

/etc/nodestat [*options*]

DESCRIPTION

writes a summary of network and hard disk activity to standard output.

OPTIONS

If no options are specified, **nodestat** returns a brief summary of network usage information for the current node.

–l	Long report–provides more information than the summary.
–config	Configuration report–displays only node-specific hardware information: CPU type, display type, etc.
–n *node_spec ...*	Provide information on specified node(s). Multiple *node_spec* strings are permitted; separate them with blanks.
–a	Report on all nodes in the network.
–r [*n*]	Repeat the **nodestat** command every *n* seconds until halted by CTRL/Q. Only counts that have changed at each iteration are displayed, and the values represent the amount of change rather than absolute values. The default value for *n* is 10 seconds.
–s [*n*]	Send *n* test messages to every node being listed (except the current node) before every repeat of the display. If this option is specified, **–r** must also be specified. This option provides a minimum amount of network activity during the wait time between **nodestat** repeats. The default value for *n* is 100 messages.
–save *pathname*	Save all statistics in the file named *pathname*.
–since *pathname*	Display counts that have changed since statistics were saved in *pathname*.

EXAMPLES

```
$ /etc/nodestat

The node ID of this node is 1FB.

**** Node 1FB ****   //diskless_$0001Fb diskless to //anger

Up since 1988/02/01 at 8:17:06 Up for 1 day 2 hours 58 mins 4 secs
Net I/O:   total=  94625   rcvs =  66912   xmits =  27713
Winchester I/O:  total=   0   reads=   0   writes=   0    {NOTE 1}
System configured with 1.0 mb of memory.
```

$ /etc/nodestat –l

```
The node ID of this node is 1FB.

**** Node 1FB ****   //diskless_$0001Fb diskless to //anger

Up since 1988/02/01 at 8:17:06 Up for 1 day 2 hours 58 mins 52 secs
Net I/O:   total=  94756   rcvs =  67010   xmits =  27746

  10436 page-in  requests issued.
  6473 page-out requests issued.
  41134 page-in  requests serviced.
  12139 page-out requests serviced.
Detected concurrency violations -- read: 0    write: 0

 Xmit count      27746          Rcv eor            0
 NACKs             272          Rcv crc          767
 WACKs            1639          Rcv timout         0
 Token inserted     65          Rcv buserr         0
 Xmit overrun        0          Rcv overrun        0

 Xmit Ack par        3          Rcv xmit-err    3042
 Xmit Bus error      0          Rcv Modem err      0
 Xmit timout        90          Rcv Pkt error     45
 Xmit Modem err      0          Rcv hdr chksum     0
 Xmit Pkt error    377          Rcv Ack par       10

    Delay switched OUT.

Winchester I/O:  total=   0   reads=  0  writes=  0  {NOTE 1}

 Not ready           0         Contrlr busy       0
 Seek error          0         Equip check        0
 Drive time out      0         Overrun            0
 CRC error percentage: 0.00%

 Last ring hardware failure detected by node 241     {NOTE 2}
  on 1988/02/02 at 10:05
System configured with 1.0 mb of memory.
A total of 0 ECCC errors were detected.
```

Notes on Examples

1. Node 1FB is running diskless, hence the absence of Winchester disk I/O activity.
2. At 10:05 A.M. on Feb. 2, 1988, the network cable was disturbed immediately upstream of node 241. This information, coupled with the network topology available from **lcnode** can help you pinpoint a hardware malfunction.

SEE ALSO

rtstat(8)

NAME

nshost – generate host tables from the name server

SYNOPSIS

nshost [**–f** *file* ...] [**–d** *domain* ...] [**–s** *server*] [**–o** *output-file*] [**–r**]

DESCRIPTION

The **nshost** command converts host and address information from name server configuration files (see **named**(8)) or from running name server(s) into the **/etc/hosts** format (see **hosts**(5)) or into the **/etc/named.hosts.rev** (reverse pointer records) format.

OPTIONS

–f *file* — Search the *file* for Standard Resource Records specifying addresses (A) or aliases (CNAME); these are then written to the standard output or the specified *output-file*. You may specify more than one **–f** *file* option.

–d *domain* — Query a running name server for the specified domain (determined by looking in the file **/etc/resolv.conf**) for this same information. You may specify more than one **–d** option. See the **–s** option.

–s *server* — Specify alternate servers. You may specify up to three **–s** options per **–d** option.

–r — Request the **/etc/named.hosts.rev** (reverse pointer records) format.

SEE ALSO

hosts(5), hostns(8), named(8);
Configuring and Managing TCP/IP.

NAME

obty – set or display the type of an object

SYNOPSIS

/etc/obty ([*object_type*] *pathname*...)

DESCRIPTION

obty is intended for system-level debugging use only. Misuse of this command can cause objects to become inaccessible and programs to behave incorrectly.

pathname (required) Specify object whose type is to be set or displayed. Wildcarding of this pathname is permitted.

object_type (optional) Specify new type setting. *object_type* must be a known type; the **lty** command lists the types currently defined on a volume.

Executable files (output of compilers and binders) are **obj**, **coff** or **unstruct**. Most other binary files are **rec**.

Default if omitted: display current type of *pathname*

EXAMPLES

The sequence of the following commands is significant.

Display current object type:

```
$ /etc/obty testfile
"testfile" object type is nil.
```

Set type to **uasc**:

```
$ /etc/obty testfile uasc
```

Display new object type:

```
$ /etc/obty testfile
"testfile" object type is uasc.
```

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/etc/ping [**–d**] [**–r**] [**–v**] *host* [*packetsize*] [*count*]

DESCRIPTION

The DARPA Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can often be difficult. The **ping** program utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (''pings'') have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of ''pad'' bytes used to fill out the packet. Default datagram length is 64 bytes, but this may be changed using the command-line option.

When using **ping** for fault isolation, you should first run it on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be ''pinged''. **ping** sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response. If you specify an optional *count*, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out (with a *count* specified), or if the program is terminated with a SIGINT, **ping** displays a brief summary.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use **ping** during normal operations or from automated scripts.

OPTIONS

Other options are:

–d Display debugging information.

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by **routed**(8C).

–v Verbose output. List ICMP packets other than ECHO RESPONSE that are received.

SEE ALSO

ifconfig(8C), netstat(1);
Configuring and Managing TCP/IP.

NAME

probenet – probe network and display error statistics

SYNOPSIS

/etc/probenet [*options*]

DESCRIPTION

This command broadcasts packets to the diagnostic socket in all nodes, then requests error counts indicating the status the broadcast was received with. It compiles counts from every node in the topology list and reports them to standard output.

OPTIONS

Use one of the following three options to specify the list of nodes to display:

–a (default) Probe all nodes responding to a **/com/lcnode** command. If the network is completely corrupted so that messages cannot make a complete pass, use one of the other two options to specify precisely which nodes to test.

–t *pathname* Probe the nodes listed in the topology file indicated. The file must contain one hexadecimal node ID per line. Any text following a space after the node ID is ignored. You can insert comment lines if they are prefixed with a "#" or "{".

–n *node_id* ... Probe the node(s) specified by the indicated hexadecimal node ID(s). A good choice of nodes to test is a set evenly spaced around the network.

Use the following options to specify which test to run:

–s *n* (default) Specify the total number of packets to be sent to each node. The default number of packets is 10. If 0 is specified for *n*, no test messages are sent, but statistics from each node are collected.

–r [*n*] Repeat the **probenet** cycle every *n* seconds. If *n* is omitted, the cycle is repeated every 10 seconds. When you press <RETURN> at the input window, the send cycle is terminated immediately and the statistics are gathered and reported.

Use the following options to specify which packets are sent:

–d *data_file* Specify that the packets are taken out of the specified data file instead of the standard built-in data pattern.

–len *n* (default)

Specify the length (in bytes) of the data portion of the test packet, in bytes. The default length is 1024 bytes.

Use the following options to control the level of detail in the statistics report.

–l Print long (detailed) error counts if there were any errors (that is, at least one transmit error (**xmit errs**) or receive error (**rcv errs**).

–err Print header for each test, but statistics only for nodes which returned errors (**xmit** and/or **rcv** errs).

–mon *fail_lim*

Print header for each pass, but statistics only on passes whose total failure count equal or exceed the *fail_lim* value.

–sens *threshold*

Open a window pane and select some output lines to append to this pane. The nodes selected are those whose error count exceed a five-node running average error count by the specified threshold value. Also, all nodes with modem errors are appended to this pane. The use of this secondary output is to do some data reduction and pick the nodes at or near points of data corruption in the network. The window pane is also stored in a named pad file called **probenet.pane**.

EXAMPLES

1. $ **/etc/probenet**

```
{Probe entire network once.  No errors detected.}

There are  5 nodes in the test.
Broadcasting 10 1024-byte packets . 85/02/20 21:16:52  # failures =  0
Last Biph hardware failure detected by node 676 on 85/02/20 at 19:15

                                      MODEM
 NODE  NAME          ATTEMPT  ERRS    ERRS   BIPH    ESB  TOKENS= 0
 ----  ------------   ------ ------ ------  ----    ---  ------
  584  *diskless          10      0      0      0      0       0  Self
   21  OS                 10      0      0      0      0       0
  AEF  BS                 10      0      0      0      0       0
   4A  HUBRIS             10      0      0      0      0       0
 3536  *diskless          10      0      0      0      0       0
```

2. $ **probenet -t node_list -s 14400 -r 3600 -d data_e3**

```
{ Probes network and displays nodes specified in file "node_list".
This node broadcasts 14400 packets in 3600 seconds, that is, four
packets per second. The packet data comes out of file "data_e3".}

There are      5 nodes in the test.
Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.
    85/02/20 21:58:19  # failures =  100
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

                                    MODEM
 NODE  NAME          ATTEMPT  ERRS   ERRS   BIPH    ESB  TOKENS= 3
 ----  ------------   ------ ------ ------  ----    ---  ------
 1967  GTX            14386      0      0      0      0      1  Self
 15F5  SWI            14386      0      0      0      0      0
 2255  BIRDIE         14384      0      0      0      0      1
  3FD  FLASH          14386      3      3      3      0      0
 2B69  STANG          14385      3      0      0      0      1

Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.
    85/02/20 21:58:41  # failures =  100
Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

                                    MODEM
 NODE  NAME          ATTEMPT  ERRS   ERRS   BIPH    ESB  TOKENS= 3
 ----  ------------   ------ ------ ------  ----    ---  ------
 1967  GTX            14383      0      0      0      0      1  Self
 15F5  SWI            14383      0      0      0      0      0
 2255  BIRDIE         14381      0      0      0      0      1
  3FD  FLASH          14383      4      4      4      0      0
 2B69  STANG          14382      4      0      0      0      1

{ Above example shows a problem between node 3FD and its predecessor
  in the network. }
```

NAME

rc, rc.local, rc.user – command scripts for auto-reboot and daemons

SYNOPSIS

/etc/rc
/etc/rc.local
/etc/rc.user

DESCRIPTION

The **rc** command script controls the automatic reboot, the **rc.local** script contains commands that are pertinent only to a specific site, and the **rc.user** script contains commands used to start Domain/OS BSD daemons, such as **mbx_helper, netman, prsvr,** and **spm.**

rc is run after an auto-reboot succeeds. **rc** starts the daemons on the system, preserves editor files, and clears the scratch directory **/tmp.**

rc.local is executed immediately before any other commands. Normally, the first commands placed in the **rc.local** file define the machine's name, using **hostname**(1).

The **rc** script invokes **rc.user** and runs it as **user.none.none**. To start up the servers listed in the **rc.user** file, uncomment the appropriate lines in that file.

NOTES

The **/etc/rc**, **/etc/rc.local**, and **/etc/rc.user** files are links, which resolve to **node_data/etc/rc**, **`node_data/etc/rc.local**, and **`node_data/etc/rc.user** respectively.

The **/etc/rc** file is owned by **root**, and protected such that only **root** can write to it.

SEE ALSO

init(8), reboot (8)

NAME

reboot – reboot the processor

SYNOPSIS

/etc/reboot [**–n**] [**–q**]

DESCRIPTION

The **reboot** command writes out cached pages to the disks and then reboots the processor.

OPTIONS

–n	Prevent the sync before rebooting.
–q	Reboot quickly, without first shutting down running processes.

NOTES

The node must be in "normal" (not service) mode when you reboot. **reboot** is not supported for **sau2** (DN3*xx*) and **sau5** (DN5*xx*, non-turbo) processors.

FILES

/sysboot
/sau*N*/aegis
/sau*N*/aegis.map

SEE ALSO

halt(8), init(8), rc(8), shutdown(8), syslogd(8) reboot(2)

NAME

renice – alter priority of running processes

SYNOPSIS

/etc/renice *priority* [[**–p**] *pid* ...] [[**–g**] *pgrp* ...] [[**–u**] *user* ...]

DESCRIPTION

renice alters the scheduling priority of one or more running processes. The *who* parameters are interpreted as process IDs, process group IDs, or usernames. Running **renice** on a process group alters the scheduling priority of all processes in the process group. Running **renice** followed by a username alters the scheduling priority of all processes owned by the user. By default, the processes to be affected are specified by their process IDs.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to PRIO_MAX (20). (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range PRIO_MIN (–20) to PRIO_MAX. Useful priorities are:

- 20 (the affected processes will run only when nothing else in the system wants to)
- 0 (the "base" scheduling priority)
- Anything negative (to make things go very fast)

OPTIONS

–g	Force *who* parameters to be interpreted as process group IDs.
–u	Force the *who* parameters to be interpreted as user names.
–p	Reset *who* interpretation to process IDs (the default).

EXAMPLE

The following example changes the priority of process IDs 987 and 32, and all processes owned by users **daemon** and **root**.

```
/etc/renice +1 987 -u daemon root -p 32
```

CAUTIONS

Non super-users can not increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

FILES

/etc/passwd to map usernames to user IDs

SEE ALSO

getpriority(2), setpriority(2)

NAME

rexecd – remote execution server

SYNOPSIS

/etc/rexecd

DESCRIPTION

rexecd is the server for the **rexec**(3X) routine. The server provides remote execution facilities with authentication based on user names and passwords.

The **rexecd** server listens for service requests at the port indicated in the ''exec'' service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

```
username too long
```

The name is longer than 16 characters.

```
password too long
```

The password is longer than 16 characters.

```
command too long
```

The command line passed exceeds the size of the argument last (as configured into the system).

```
Login incorrect
```

No password file entry for the user name existed.

```
Password incorrect
```

The wrong was password supplied.

```
No remote directory
```

The **chdir** command to the home directory failed.

```
Try again
```

A **fork** by the server failed.

```
<shellname>: ...
```

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

CAUTIONS

Indicating ''Login incorrect'' as opposed to ''Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

SEE ALSO

rexec (3X)

NAME

/etc/rgy_admin – registry server administrative tool

SYNOPSIS

/etc/rgy_admin

DESCRIPTION

The **rgy_admin** tool administers registry servers. It can view or modify the registry replica list, reinitialize replicas, delete replicas, stop servers, and change the registry master site.

Note that **rgy_admin** cannot add, delete, or modify data entries contained in the registry database, such as names and accounts; use **edrgy** to perform these tasks. To create a registry replica or to restart a server, use **rgyd**, the registry daemon.

Once invoked, **rgy_admin** enters an interactive mode in which it accepts the commands described in the next section.

COMMANDS

Most **rgy_admin** commands operate on a default host. You use the **set** command to establish the default host, which is remembered until changed by another **set**. In the following command descriptions, we identify the default host as *default_host*. If a command operates on a host other than the default, we identify this host as *other_host*.

Several of the **rgy_admin** commands require you to set the default host to the master registry site.

The host name you supply as a *default_host* or *other_host* takes the form *family*:*host* or *host*. The only currently supported *family* is **dds**, the Domain protocol family. You can specify a host in this family by its entry directory or by its network address. For example, **dds://clara**, **//clara**, **dds:#1234.abcd**, and **#1234.abcd** are all acceptable host names.

become [**–master**] [**–slave**] [**–ro** | **–wr**]

The **–master** option causes the replica at *default_host* (which must be a slave replica) to become the master. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–slave** option causes the replica at *default_host* (which must be the master replica) to become a slave. This operation can cause updates to be lost; the **change_master** command is the preferred means of designating a different master replica.

The **–ro** option makes the replica list read-only. The *default_host* must be set to the master registry site.

The **–wr** option makes the replica list writable. The *default_host* must be set to the master registry site.

change_master –to *other_host*

Change the master replica of the registry from *default_host* to *other_host*. The *default_host* must be set to the master registry site.

The current master server copies its database to the replica at *other_host*, becomes a slave, then tells the replica at *other_host* to become the master.

delrep *other_host* [**–force**]

Delete the registry replica at *other_host*. The *default_host* must be set to the master registry site.

The master server marks the replica at *other_host* as deleted and propagates the deletion to all other replicas on its list. When it has actually delivered the delete request to the replica at *other_host*, the master server removes that replica from its own replica list.

The **–force** option causes a more drastic delete. It deletes *other_host* from the replica list at the master registry, which then propagates the delete request to the replicas at the hosts that remain on its list. Since this operation never communicates with the deleted replica, you should use **–force** only when the replica has died irrecoverably. If you use **–force** while the replica at *other_host* is still running, you should then **reset** the deleted replica.

help List the **rgy_admin** commands and show their allowed abbreviations.

info Get status information about the replica at *default_host*.

initrep *other_host*

Reinitialize the registry server at *other_host*. The *default_host* must be set to the master registry site. The *other_host* must be a slave site.

The master registry copies its entire database (or that of another up-to-date replica) to the replica at *other_host*.

lrep [**–state**] [**–na**]

List the registry replica sites as stored in the replica list at *default_host*.

The **–state** option shows the current state and update time on each host.

The **–na** option shows the network address of each host.

monitor [**–r** *m*]

Periodically list the registry replica sites as stored in the replica list at *default_host* and show the current state and update time at each site.

The **–r** option causes the sites to be listed every *m* minutes. If you omit this option, the period is 15 minutes.

quit Quit the **rgy_admin** session.

reprep *other_host*

Replace the network address for *other_host* in the registry replica list. The *default_host* must be set to the master registry site.

The master replica propagates the new network address for *other_host* to all other registry replica lists. Use this command only if a replica site's network number changes.

reset *other_host*

Reset the registry replica at *other_host*. The registry server at *other_host* deletes its copy of the registry and stops running. This command does not delete *other_host* from any replica lists.

set [**–h** *host_name* | **–m**]

Set the default host. Subsequent commands that do not specify a host will go to this host.

The **–h** option specifies a replica to use as the default.

The **–m** option causes the master replica to be the default.

With no options, **set** locates a registry replica and sets it as the default.

site [*host_name*]

If *host_name* is specified, the command sets the default host.

If *host_name* is not specified, the command gets status information about *default_host*.

state –in_maintenance | –not_in_maintenance

Put the master registry server into maintenance state or take it out of maintenance state. The *default_host* must be set to the master registry site.

With the **–in_maintenance** flag, **state** causes the master registry server to save its database onto disk and refuse any updates.

With the **–not_in_maintenance** flag, **state** causes the master registry server to reload its database from disk, return to its normal "in service" state, and (if its database and/or replica list are writable) start accepting updates.

stop Stop the registry server that is running at *default_host*.

EXAMPLES

Start **rgy_admin**, list the replicas and their states, then set the default host to the master replica:

```
$ /etc/rgy_admin
Default object: rgy  default host: dds://george
State: in service  slave
rgy_admin: lrep -st
  (master) dds://martha  state: in service  1987/11/16.12:46:59
           dds://george  state: in service  1987/11/16.12:46:59
           dds://thomas  state: in service  1987/11/16.12:46:59
rgy_admin: set -m
          Default object: rgy  default host: dds://martha
          State: in service  master  replica list is writeable
```

NAME

rgy_create – registry creation utility

SYNOPSIS

rgy_create

DESCRIPTION

The **rgy_create** tool creates a new registry database on the local node, initialized with reserved names and accounts. It ordinarily should be run only once at a site. Replicas of the database are created by running **rgyd** with the **–create** option.

You must be root to invoke **rgy_create**.

Note that to convert a pre-SR10 registry to SR10 format, you should run only the **cvtrgy** tool, and you will never need to use **rgy_create**.

NAME

rgy_merge – merge registry database

SYNOPSIS

rgy_merge **–from** *//site* [{ **–merge** | **–compare** } **–verbose**]

DESCRIPTION

The **rgy_merge** utility merges the contents of two registry databases, a source database and a target database. You typically use it when you are joining two networks that have been operating separately and you want to combine their registry databases.

You must invoke **rgy_merge** while logged in as root at the master registry node for the target registry. Use the required **–from** *//site* argument to specify the master registry node for the source registry. The merge takes less time if the source database is smaller than the target database.

After you invoke **rgy_merge**, the tool prompts you to "login" with an account that owns the source registry.

Without a **–merge** or **–compare** option, **rgy_merge** attempts to add each entry in the source database to a copy of the target database and reports any conflicts in names or UNIX numbers. If there are no conflicts or errors, the tools asks whether you want to actually update the target database. If you respond affirmatively, it performs the merge on the target database and makes all replicas of the source registry slave replicas of the target registry.

If you specify **–merge**, **rgy_merge** performs the merge without querying you, provided there are no conflicts or errors. If you specify **–compare**, **rgy_merge** only checks for conflicts and does not perform a merge even if there are none.

The **–verbose** option causes **rgy_merge** to generate a verbose transcript of its activity.

NAME

rgyd – network registry server

SYNOPSIS

/etc/rgyd [–create|–recreate|–restore_master]

DESCRIPTION

rgyd is the network registry daemon. It manages all access to the network registry database. You must be the super-user to invoke **rgyd**.

The daemon can be replicated, so that several copies of the database exist on a network or an internet, each managed by a **rgyd** process. Only one registry daemon, the master, can accept operations that change the database (such as adding an account). If the daemon is replicated, the other replicas are slaves, which accept only lookup operations (such as validating a login attempt).

A Local Location Broker daemon (**llbd**) must be running on the local node when **rgyd** is started. Typically, both daemons are started at boot time from **/etc/rc**. The server will place itself in the background when it is ready to service requests.

OPTIONS

–create Create a replica of the network registry. This option creates a copy of the registry database and starts a slave server process. You use **–create** only the first time you start a slave server process on a node. When you restart the daemon, you do not need any options at all. To create the master replica, use either **cvtrgy** (if you are converting an SR9 registry to SR10 format) or **rgy_create** (if you are creating a new SR10 registry).

–recreate Recreate a slave replica. You should use this option only if a slave's copy of the database has been irreparably corrupted. It destroys the existing database and creates a new one.

–restore_master Restart a master server and reinitialize all slave replicas. You should use this option only to recover from a catastrophic failure of the master node, (for example, if the database has been corrupted and then restored from a backup tape).

EXAMPLES

All of the commands shown in these examples must be run by root.

1. Start the master replica of the registry after you have created the master database via **rgy_create** or **cvtrgy**:

   $ **/etc/server –p /etc/rgyd**

2. Start a slave replica of the registry.

   $ **/etc/server –p /etc/rgyd –create**

3. Restart an existing replica (master or slave) of the registry.

   $ **/etc/server –p /etc/rgyd**

4. Restart an existing replica of the registry on the remote host //yak.

   $ **/etc/crp –on //yak –cps –n rgyd //yak/etc/rgyd**

SEE ALSO

cvtrgy(1), rgy_admin(8), rgy_create(8), rgy_merge(8)

NAME

rlogind – remote login server

SYNOPSIS

/etc/rlogind [–d]

DESCRIPTION

rlogind is the server for the **rlogin**(1C) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

rlogind listens for service requests at the port indicated in the "login" service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**(3N), **hosts**(5) and **named**(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, **rlogind** allocates a pseudoterminal (see **pty**(4)), and manipulates file descriptors so that the slave half of the pseudoterminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in **rshd**(8C), but if automatic authentication fails, it reprompts the user to log in as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudoterminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty**(4) is invoked to provide CTRL/S and CTRL/Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM"; see **environ**(7). The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudoterminal.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

`Try again.`

A **fork** by the server failed.

`/bin/sh: ...`

The user's login shell could not be started.

CAUTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol should be used.

NAME

rmt – remote magtape protocol module

SYNOPSIS

/etc/**rmt**

DESCRIPTION

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an **rexec**(3X) or **rcmd**(3X) call.

The **rmt** program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number*\n

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number*\n*error-message*\n,

where *error-number* is one of the possible error numbers described in **intro**(2) and *error-message* is the corresponding error string as printed from a call to **perror**(3).

COMMANDS

The protocol is comprised of the following commands (a space is present between each token).

O *device mode*	Open the specified *device* using the indicated *mode*. *Device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to **open**(2). If a device had already been opened, it is closed before a new open is performed.
C *device*	Close the currently open device. The *device* specified is ignored.
L *whence offset*	Perform an **lseek**(2) operation using the specified parameters. The response value is that returned from the **lseek** call.
W *count*	Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the **write**(2) call.
R *count*	Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. **rmt** then performs the requested **read**(2) and responds with A*count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.
I *operation count*	Perform a MTIOCOP **ioctl**(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of

the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the **ioctl** call. The return value is the *count* parameter when the operation is successful.

S Return the status of the open device, as obtained with a MTIOCGET **ioctl** call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes **rmt** to exit.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

rcmd(3X), rexec(3X), mtio(4), rdump(8C), rrestore(8C)

CAUTIONS

People tempted to use this for a remote file access protocol are discouraged.

NAME

route – manually manipulate the routing tables

SYNOPSIS

/etc/route [**–f**] [**–n**] [*command args*]

DESCRIPTION

route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, **routed**(8C), should tend to this task.

route accepts three commands: **add**, to add a route; **addp**, to add a priority route; and **delete**, to delete a route.

The **addp** command adds a priority route. The TCP/IP server process will use priority routes before default routes or routes established by **routed**(8C). A route added with **addp** will appear first in the gateway table (displayed with the BSD command **netstat –r**(1)). You can only add priority routes with **addp**. Routes added manually by **route** cannot be deleted by **routed**(8C).

COMMAND SYNTAX

All commands have the following syntax:

/etc/route *command* [**net** | **host**] *destination gateway* [*metric*]

where *destination* is the destination host or network, *gateway* is the next-hop gateway to which packets should be addressed, and *metric* is a count indicating the number of hops to the *destination*. The metric is required for **add** and **addp** commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. The optional keywords **net** and **host** force the destination to be interpreted as a network or a host, respectively. If the *destination* has a "local address part" of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

If the route is to a destination connected through a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname**(3N). If this lookup fails, **getnetbyname**(3N) is then used to interpret the name as that of a network.

You can add a default route as follows:

/etc/route *add default gateway_name* [*non-zero metric*]

TCP/IP software will use the default route when other routes occuring earlier in the routing table have failed, or when there are no other possible routes.

route uses a raw socket and the SIOCADDRT and SIOCDELRT **ioctl**'s(2) to do its work. As such, only the super-user may modify the routing tables.

OPTIONS

–f "Flush" the routing tables of all gateway entries. Using this option in conjunction with one of the commands described above flushes the tables prior to the command's application.

–n Suppress printing symbolic host and network names when reporting actions.

DIAGNOSTICS

```
add [ host | network ] %s: gateway %s flags %x
```

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call. If the gateway address used was not the primary address of the gateway (the first one returned by **gethostbyname**(3N)), the gateway address is printed numerically as well as symbolically.

```
delete [ host | network ] %s: gateway %s flags %x
```

As above, but when deleting an entry.

```
%s %s done
```

When the **–f** flag is specified, each routing table entry deleted is indicated with a message of this form.

```
Network is unreachable
```

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

```
not in table
```

A delete operation was attempted for an entry that wasn't present in the tables.

```
routing table overflow
```

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

intro(4N), routed(8C);
Configuring and Managing TCP/IP.

NAME

routed – network routing daemon

SYNOPSIS

/etc/routed [**–g**] [**–s**] [**–q**] [**–t**] [**–n**] [**–f**] [**–h**] [*logfile*]

DESCRIPTION

The **routed** daemon is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries. It uses a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation **routed** listens on the **udp**(4P) socket for the **route** service (see **services**(5)) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the SIOCGIFCONF **ioctl**(2) to find those directly connected interfaces configured into the system and marked ''up'' (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. **routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a ''hop count'' metric (a count of 16, or greater, is considered ''infinite''). The metric associated with each route returned provides a metric *relative to the sender*.

Response packets received by **routed** are used to update the routing tables if one of the following conditions is satisfied:

1) No routing table entry exists for the destination network or host, and the metric indicates the destination is ''reachable'' (i.e. the ''hop count'' is not infinite).

2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; to decide this, the metric of the new route is compared against the one stored in the table.

When an update is applied, **routed** records the change in its internal tables. The change is reflected in the next *response* packet sent.

In addition to processing incoming packets, **routed** also checks the routing table entries periodically. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to ensure that the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

OPTIONS

routed supports several options:

–g This flag is used on internetwork routers to offer a route to the ''default'' destination. This option is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

–s Forces **routed** to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

–q This option is the opposite of the –s option.

–t If the **–t** option is specified, all packets sent or received are printed on the standard output. In addition, **routed** will not divorce itself from the controlling terminal, so that interrupts from the keyboard will kill the process.

–d Not supported by Domain/OS BSD.

Domain/OS BSD EXTENSIONS

–n Directs **routed** not to install changes into the local routing table. However, the **routed** process continues to receive broadcasts from other **routed** processes. The **–n** option is used for debugging purposes.

–f Directs **routed** at startup to ''flush'' (purge) all routes from the local routing table, except routes added manually with **/etc/route**(8C).

–h Exit, if not supplier, when routing table is stable. Use this switch on hosts only, not on gateways.

Any other argument supplied is interpreted as the name of the file in which **routed**'s actions should be logged. This log contains information about any changes to the

routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, **routed** supports the notion of distant passive and active gateways. When **routed** is started up, it reads the file **/etc/gateways** to find gateways that may not be located using only information from the SIOGIFCONF **ioctl**(2). Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a **routed** process running on the machine).

Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. External gateways are also passive, but they are not placed in the routing table nor are they included in routing updates The function of external entries is to inform **routed** that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The **/etc/gateways** file is comprised of a series of lines, each in the following format:

< **net** | **host** > *name1 gateway name2 metric value* < **passive** | **active** | **external** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

Name1 The name of the destination network or host. This may be a symbolic name located in **/etc/networks** or **/etc/hosts** (or, if started after **named**(8), known to the name server), or an Internet address specified in ''dot'' notation; see **inet**(3n).

Name2 The name or address of the gateway to which messages should be forwarded.

Value A metric indicating the hop count to the destination host or network.

One of the keywords **passive**, **active** or **external** indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the **routed** protocol.

Internetwork routers that are directly attached to the ARPANET or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather then using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the Computer Systems Research Group at Berkeley.

For a node to run **routed**, it must be correctly configured to run BSD TCP/IP. See *Configuring and Managing TCP/IP* for more information about **routed**.

NOTES

The **routed** daemon is normally started on a node at boot time, from the **/etc/rc.local** file. We recommend that you run **routed** on each gateway to dynamically update the gateway's routing tables. You can also run **routed** on hosts so they receive the latest routing information.

FILES

/etc/gateways	for distant gateways

BUGS

The kernel's routing tables may not correspond to those of **routed** when redirects change or add routes. The only remedy for this is to place the routing process in the kernel.

routed does not incorporate other routing protocols, such as Xerox NS and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

routed does not currently listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

SEE ALSO

udp(4P), htable(8), route(8C), rc(8);
Configuring and Managing TCP/IP;
"Internet Transport Protocols", XSIS 028112, Xerox System Integration Standard.

NAME

rshd – remote shell server

SYNOPSIS

/etc/rshd

DESCRIPTION

rshd is the server for the **rcmd**(3X) routine and, consequently, for the **rsh**(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

rshd listens for service requests at the port indicated in the ''cmd'' service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**(3N), **hosts**(5) and **named**(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client 's machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server 's machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) **rshd** then validates the user according to the following steps. The local (server-end) user name is looked up in the password file and a **chdir** is performed to the user's home directory. If either the lookup or **chdir** fail, the connection is terminated. If the user is not the super-user, (user id 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered ''equivalent''. If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and

identity of the user on the client's machine. If this lookup fails, the connection is terminated.

9) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the execution of the login shell).

```
locuser too long
```

The name of the user on the client's machine is longer than 16 characters.

```
remuser too long
```

The name of the user on the remote machine is longer than 16 characters.

```
command too long
```

The command line passed exceeds the size of the argument list (as configured into the system).

```
Login incorrect.
```

No password file entry for the user name existed.

```
No remote directory.
```

The **chdir** command to the home directory failed.

```
Permission denied.
```

The authentication procedure described above failed.

```
Can't make pipe.
```

The pipe needed for the **stderr**, wasn't created.

```
Try again.
```

A **fork** by the server failed.

```
<shellname>: ...
```

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

CAUTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

SEE ALSO

rsh(1C), rcmd(3X);
Configuring and Managing TCP/IP.

NAME

rtchk – test traffic between adjacent routers

SYNOPSIS

/etc/rtchk [*options*]

DESCRIPTION

rtchk performs a simple test to verify that the router is able to pass packets to and from an adjacent router. **rtchk** is for use in a Domain internet.

Use the **–device** option to specify a network controller to test. You must give a device type (for example, RING, IIC) to the device option. The **rtsvc** program, with no command-line options, shows you which network devices your node has.

Older versions of **rtchk** used a different command-line syntax to specify the type of network hardware checked. The old command-line options still work in **rtchk** version 10.1, but are no longer supported.

For more information on **rtchk**, see *Managing Domain Routing and Domain/OS in an Internet*.

OPTIONS

–n *net.node_id*
: Test packet transmission to and from the specified node. The network id *net* must be a network that the router touches. If you use **–n** without **–dev**, you must specify a *net.node_id*. If you use **–n** with **–dev**, you must specify only the *node_id*, without the network number.

–dev[ice] *dev-name* [*dev-num]*
: Test packet transmission over a specific network device. Use the **rtsvc** program to display the names (used for *dev-name*) and controller numbers (used for *dev-num*) of the network devices attached to your node.

  If you do not also specify a **–n** option, **rtchk** broadcasts its test packets. If the network contains more than one other node, **rtchk** receives more responses to its test packets than expected and prints warning messages. If you specify a **–n** option with the **–dev** option, **rtchk** sends the test packets only to the node you specify.

–s *n*
: Specify the number of test packets to exchange with the other router. If **–s** is not specified, 10 packets are exchanged.

–dat (default)
: Specify that each test packet carries 1024 bytes of test data.

–nodat
: Omit test data from the test packets.

EXAMPLES

Exchange 1000 test packets with node 4851 on network 3CE02A8. The router must be attached to that network.

$ **/etc/rtchk –n 3CE02A8.4851 –s 1000**

Exchange 10 short test packets with the other node attached to the IIC or T1 connection.

$ **/etc/rtchk –nodat**

Exchange 100 test packets with the other node on the IIC or T1 network.

$ **/etc/rtchk –dev iic –s 100**

Exchange 10 test packets with node 666 on the ring network.

$ **/etc/rtchk –dev ring –n 666**

NAME

rtstat – display internet router information

SYNOPSIS

/etc/rtstat [*options*]

DESCRIPTION

rtstat shows the behavior of an internet router at each of its network ports. **rtstat** is used in Domain internets. However, it can provide information about non-routing nodes as well as routing nodes.

For more information on **rtstat**, see *Managing Domain Routing and Domain/OS in an Internet*.

OPTIONS

–dev Report device-specific statistics for each port.

–net [*net_id ...*]
Report counts of references to each network specified. The reference counts for each network are roughly proportional to the number of packets transmitted towards the network, but may be somewhat higher. **–net** with no arguments uses the list of visible networks.

–r [*n*] Repeat every *n* seconds. If *n* is omitted, repeat every 10 seconds.

–n *node_spec ...*
Report statistics for each node in the list.

If this option is omitted, **rtstat** reports statistics for the local node only.

–desc[ribe] Print a description, several lines long, of each statistic. The description appears only once for each statistic, the first time it is printed with a nonzero value.

EXAMPLES

1. $ **/etc/rtstat**

```
------------------------------------------------------------
1232.3D9          pkts routed:   110024    queue oflo:       0
                  misrouted:          0    rt too far:      14

     RING         pkts sent:      73278    pkts rcvd:    72434

     IIC          pkts sent:      67830    pkts rcvd:    61077
```

2. $ **/etc/rtstat -net**

```
------------------------------------------------------------------
1232.3D9            pkts routed:   110024    queue oflo:        0
                    misrouted:          0    rt too far:       14

     RING           pkts sent:      73278    pkts rcvd:     72434
                    towards net:     1232    ref cnt:       74540

     IIC            pkts sent:      67830    pkts rcvd:     61077
                    towards net:     1234    ref cnt:       53532
                    towards net:     1231    ref cnt:        9193
                    towards net:     1233    ref cnt:        5105
```

NAME

rtsvc – set or display internet routing service

SYNOPSIS

/etc/rtsvc [–device *dev-name* [*dev-number*] [*options*]]

DESCRIPTION

rtsvc displays or alters the characteristics of a network port. **rtsvc** is used in Domain internets. You must be logged on to the node you wish to control in order to use **rtsvc**.

For complete information on **rtsvc**, see *Managing Domain Routing and Domain/OS in an Internet.*

OPTIONS

If no options are specified, **rtsvc** displays the characteristics of every active network. If you specify any other options, you must specify the type of network controller by using a **–device** command-line option.

You may use only one **–device** option on any command line:

–dev[ice *dev-name* [*dev-number*]

Specify the network device type: RING, IIC, or USER (for EtherBridge routers). The device number applies only to USER devices. You may use the name ETHERBRIDGE in place of USER if you prefer. The *dev-number* option applies only to USER networks, and is required. Find the device number by using **rtsvc** without command line options (as shown in the examples).

Earlier versions of the **rtsvc** command used a different command-line syntax for specifying network devices. The old command lines still work, but you should start using the new **–device** command lines as soon as possible. Future versions of **rtsvc** will not accept the older command lines.

This option changes the network ID of any network port:

–net *net_id* Assign the port a hexadecimal network ID number.

Note: If you use this option to change the network ID of a port on an active router, other nodes on the network can stop communicating with each other. Use this option only as directed in *Managing Domain Routing and Domain/OS in an Internet.*

You can specify only one of the following options at a time:

–route Allow routing service to or from the port.

–noroute Allow normal Aegis requests but no routing service.

–off Do not allow Aegis requests or routing service.

–user *nn* Set an EtherBridge network. The value is not changed until the routing node is rebooted or the routing process is stopped and restarted.

EXAMPLES

$ **/etc/rtsvc –device iic –net 007302ED –route**

Assign a network ID to the Interphase controller and allow internet routing at that port.

$ **/etc/rtsvc –dev ring –noroute**

Stop internet routing through the ring port, but allow normal Aegis requests for paging, file service, etc. Do not change the node's network ID:

$ **/etc/rtsvc**

Display the networks attached to this node.

```
    Controller          Net ID      Service offered
==================    ========    ====================
RING                       76A0    Own traffic only
USER             46        768C    Port not open
```

The node in the last example touches two networks: a Domain ring and an ETHERNET, via the EtherBridge product. You need the device number information ("46") from this display in order to turn on routing at the EtherBridge network. Use the device number as shown here:

$ **/etc/rtsvc –dev user 46 –route**

"46" is the device number.

NAME

rwhod – system status server

SYNOPSIS

/etc/rwhod [**–w**]

DESCRIPTION

The **rwhod** server maintains the database used by the **rwho**(1C) and **ruptime**(1C) programs. Its operation is predicated on the ability to broadcast messages on a network.

On Apollo networks, **rwhod** has been changed to take advantage of the Domain distributed file system and to reduce contention for the **rwho** directory. Since on Apollo networks, there is no need for every host on the network to maintain its own **/usr/spool/rwho** directory, a change has been made to allow you to specify one host per network who will write the information to a master **/usr/spool/rwho** directory. All the other hosts link to that master directory to access the information but do not write to the directory. Specifically, **rwhod** writes to **/usr/spool/rwho** only if that directory is local to the node, or if the optional –w switch is supplied.

rwhod both produces and consumes system status information. It periodically queries the state of the system and constructs status messages which are broadcast on a network, and it listens for other **rwhod** servers' status messages as well. When it receives a status message from another server, **rwhod** validates it and records it in a file located in the directory **/usr/spool/rwho**, if the directory is physically located on the host or if **rwhod** was started with the –w switch. On hosts where **rwhod** was started without the –w switch and where **/usr/spool/rwho** is a link to a remote host, **rwhod** does not write to the directory.

The **rwho** server transmits and receives messages at the port indicated in the ''rwho'' service specification. The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];/* tty name */
        char    out_name[8];/* user id */
        long    out_time;/* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
```

```
        int     we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before transmission. The load averages represent averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **gethostname**(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes an entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at an **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod.hostname** in the directory **/usr/spool/rwho**. These files contain only the most recent message, in the format described above.

OPTIONS

–w Update the files in the **/usr/spool/rwho** directory, even if the directory is not local to the node.

NOTES

Note that **rwhod** will update the files in the **/usr/spool/rwho** directory in two cases:

1. If the **/usr/spool/rwho** directory is physically located on that host.

2. If **rwhod** was started with the **–w** switch.

To reduce contention for the rwho directory and provide updated information to hosts, we recommend that you

- Physically locate the **/usr/spool/rwho** directory on the network's TCP/IP administrative node. All other hosts on the network should be set up so that **/usr/spool/rwho** is a link to the administrative node.
- If you are running **rwhod** in a Domain internet, you can configure your network in one of two ways:

  1. Configure one node on each subnet to have a local **/usr/spool/rwho** and link all other nodes on that subnet to that node.

  2. Configure all nodes in the internet to link to one master TCP/IP administrative node containing the **/usr/spool/rwho** directory. This enables the **rwho** and **ruptime** utilities to see all nodes in the internet.

In order to make this arrangement work, you must run **rwhod** –w on one node on each subnet. They will then write their subnet's information to the directory on the internet's master TCP/IP administrative node.

You also can run **rwhod** on a gateway to see broadcasts from two networks.

rwhod, like other BSD daemons, is invoked at boot time by the **/etc/rc.local** startup file. See *Configuring and Managing TCP/IP* for more information about **rwhod**.

NOTES

People often interpret the server's dying as a machine in the network going down.

SEE ALSO

rwho(1C), ruptime(1C), rc(8)

NAME

sa, accton – system accounting

SYNOPSIS

/etc/sa [**–abcdDfijkKlnrstuv**] [**–S** *savacctfile*] [**–U** *usracctfile*] [*file*]

/etc/accton [*file*]

DESCRIPTION

With an argument naming an existing *file*, **accton** causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

The **sa** command reports on, cleans up, and generally maintains accounting files.

sa is able to condense the information in **/usr/adm/acct** into a summary file **/usr/adm/savacct** which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system **/usr/adm/acct** can grow by 100 blocks per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; **/usr/adm/acct** is the default.

Output fields are labeled ''cpu'' for the sum of user+system time (in minutes), ''re'' for real time (also in minutes), ''k'' for CPU-time averaged core usage (in 1kKunits), ''avio'' for average number of I/O operations per execution. With options fields labeled ''tio'' for total I/O operations, ''k*sec'' for CPU storage integral (kilo-core seconds), ''u'' and ''s'' for user and system CPU time alone (both in minutes) will sometimes appear.

OPTIONS

–a Print all command names, even those containing unprintable characters and those used only once. By default, those are placed under the name '***other.'

–b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.

–c Besides total user, system, and real time for each command print percentage of total time over all commands.

–d Sort by average number of disk I/O operations.

–D Print and sort by total number of disk I/O operations.

–f Force no interactive threshold compression with –v flag.

–i Don't read in summary file.

–j Instead of total minutes time for each category, give seconds per call.

–k Sort by CPU-time average memory usage.

–K Print and sort by CPU-storage integral.

–l Separate system and user time; normally they are combined.

–m Print number of processes and number of CPU minutes for each user.

–n Sort by number of calls.

–r Reverse order of sort.

–s Merge accounting file into summary file **/usr/adm/savacct** when done.

–t For each command report ratio of real time to the sum of user and system times.

–u Superseding all other flags, print for each command in the accounting file the user ID and command name.

–v Followed by a number *n*, types the name of each command used *n* times or fewer. Await a reply from the terminal; if it begins with 'y', add the command to the category '**junk**.' This is used to strip out garbage.

–S *savacctfile* The following filename is used as the command summary file instead of **/usr/adm/savacct.**

–U *usracctfile*
The following filename is used instead of **/usr/adm/usracct** to accumulate the per-user statistics printed by the –m option.

FILES

/usr/adm/acct	raw accounting
/usr/adm/savacct	summary
/usr/adm/usracct	per-user summary

NOTES

Domain/OS systems report half of the process time as user time, and half as system time. Add these two to obtain the overall process CPU time when using the **-l** option.

The **/usr/adm** directory is usually a link to '**node_data/adm**. This allows you to have a separate accounting file for each node, if you run accounting on diskless nodes.

SEE ALSO

acct(2)

NAME

salacl – salvage an access control list

SYNOPSIS

/etc/salacl [*options*] [*volume*]

DESCRIPTION

salacl salvages the (Access Control List (ACL) structure on the volume you specify. It merges duplicate ACLs into a single copy, and deletes unused ACLs. You should run **salacl** once a week or so unless you never receive reports that reference counts need repairing (that is, everything is perfect).

salacl cannot merge duplicate ACLs on files that are currently in use (locked) (for example, library files). This is not especially serious, as the duplication consumes very little disk space. To merge all possible ACLs on a node, including things like libraries, bring the node up diskless, mount the volume using **mtvol**, and then run **salacl** on the mounted volume.

volume (optional) Specify the entry directory pathname for the volume whose acls you intend to salvage. Note that **salacl** cannot salvage a volume mounted on a remote node.

Default if omitted: "/" node entry directory

OPTIONS

–n[o_]s[ummary] Suppress summary information.

–v[erify] Verify only; do not delete or merge any ACLs.

–n[o_]m[erge] Do not merge duplicate ACLs into a single copy.

–l[ist] List the UIDs of the ACLs that are being combined.

–u This option causes **salvol** to scan for files and directories with ACLs. When it encounters one, it puts out a line with UID of the ACL, followed by the UID of the file or directory. It also indicates the type of ACL, whether file or directory, and whether initial or not.

EXAMPLES

```
$ /etc/salacl
Warning: unable to merge two equivalent ACLs:
//grover/sys/node_data/boot_shell -
object is in use  (OS/file server)
acl objects found:                  24
acls merged with equivalents:       12
```

Salvage the ACLs of the current volume.

```
$ /etc/salacl -v
acl objects found:                 24
acls which could be merged:        12

$ /etc/salacl -l
3A39A9F3.4000CBD6 39A77EF0.4000CBD6 (equiv acl)
3A75D6C2.E000CBD6 39A77EF0.4000CBD6 (equiv acl)
3AEE67CA.7000CBD6 39A77EF0.4000CBD6 (equiv acl)
3AEF69EE.E000CBD6 39A77EF0.4000CBD6 (equiv acl)
3AEFA618.7000CBD6 382E6E87.A000CBD6 (equiv acl)
ACL objects found:                 43
ACLs merged with equivalents:       5

$ /etc/salacl -v -u
3AE41FE2.D000CBD6 393D4261.B000CBD6 (dir)
3AE41F4B.5000CBD6 380A1479.1000CBD6 (dir)
3A39A9F3.4000CBD6 3A39A9F2.3000CBD6 (file)
3AE41F85.0000CBD6 380A1472.F000CBD6 (dir)
3AE41F84.F000CBD6 380A1472.F000CBD6 (dir def)
3AE41F81.E000CBD6 380A1472.F000CBD6 (file def)
```

NAME

salvol – verify and correct allocation of disk blocks

SYNOPSIS

from shell: /**etc/salvol** [*options*] [*lv_num*]
from mnemonic debugger: **ex salvol**

DESCRIPTION

Each logical volume is divided into disk blocks. **salvol** verifies and, if necessary, corrects the tables that describe the allocation of disk blocks to the files stored on the disk. **salvol** also returns to the free space pool all blocks that are no longer in use: those allocated to temporary files or to deleted portions of permanent files.

salvol can usually restore a disk after a system crash or an improper unmounting of a volume.

If no options are specified on the command line, then **salvol** prompts for all required arguments and options.

lv_num (optional)
A decimal value for which logical volume number to salvage.

OPTIONS

–a Read all blocks in all files. This option will take longer to run and is useful for finding block header errors or file blocks that cannot be read. This option is not useful for finding multiply allocated blocks or general disk problems.

–c *c*[*cnum*:[*d_num*]
controller type:
c = {w, s, f} (for winchester, storage module or floppy)
cnum = controller number, if specified, must be followed by a ':'
d_num = drive number

This flag must be specified if any command line options are used.

–f Fix error without prompting. (The default is to prompt.)

–h Print help text.

–n Check disk; only salvage if disk needs salvaging. This option is useful in scripts that mount secondary disks at boot time.

–p Polite mode; pause before overflowing screen. (The default is offline.)

–s Show some file statistics at completion.

–t Terminal mode, do not pause during output. (The default is online.)

–v Verify only; do not write anything to disk.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/lib/sendmail [**flags**] [*address ...*]

newaliases

mailq [**–v**]

DESCRIPTION

sendmail sends a message to one or more recipients, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not a user interface routine; other programs provide simpler front ends. **sendmail** is used only to deliver pre-formatted messages.

With no flags, **sendmail** reads its standard input up to an end-of-file, or to a line consisting of a single dot, and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally, the sender is not included in any alias expansions. For example, if ''john'' sends to ''group'' and ''group'' includes ''john'' in the expansion, then the letter will not be delivered to ''john''.

In aliases, the first character of a name may be a vertical bar, to cause **sendmail** to interpret the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep **sendmail** from suppressing the blanks between arguments. Aliases may also have the syntax ''**:include:** *filename* '' to ask **sendmail** to read the named file for a list of recipients (see EXAMPLES, below).

FLAGS

–ba Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the ''From:'' and ''Sender:'' fields are examined for the name of the sender.

–bd Run as a daemon. This requires Berkeley IPC. **sendmail** will fork and run in background, listening on socket 25 for incoming SMTP connections. This is normally run from **/etc/rc**.

–bi Initialize the alias database.

–bm Deliver mail in the usual way (default).

–bp Print a listing of the queue.

–bs Use the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the **–ba** flag that are compatible with SMTP.

–bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

–bv Verify names only – do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

–bz Create the configuration freeze file. Domain/OS BSD doesn't support frozen configuration files.

–C*file* Use alternate configuration file. **sendmail** refuses to run as root if an alternate configuration is specified. The frozen configuration file is bypassed.

–d*X* Set debugging value to *X*.

–F*fullname* Set the full name of the sender.

–f*name* Set the name of the ''from'' person (i.e., the sender of the mail). **–f** can only be used by ''trusted'' users (normally *root, daemon,* and *network*), or if *name* is the same as your log-in name.

–h*N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, ''Received:'' lines in the message are counted.

–n Don't perform aliasing.

–o*x value* Set option *x* to the specified *value*. Options are described below.

–q[*time*] Processed saved messages in the queue at *time* intervals. If *time* is omitted, process the queue once. *Time* is given as a tagged number, with ''s'' being seconds, ''m'' being minutes, ''h'' being hours, ''d'' being days, and ''w'' being weeks. For example, ''–q1h30m'' or ''–q90m'' would both set the interval between processing passes to one hour thirty minutes. If *time* is specified, **sendmail** will run in background. The option can be used safely with **–bd.**

–r*name* An alternate (obsolete) form of the **–f** flag.

–t Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed; that is, they will not receive copies even if listed in the message header.

–v Go into verbose mode. Alias expansions will be announced, etc.

OPTIONS

A number of processing options may be set. Normally, a system administrator will set these. Options may be invoked either on the command line using the **–o** flag or in the configuration file. The options are:

A*file*	Use an alternate alias file.
c	Queue messages, rather than connecting immediately to mailers that are considered ''expensive''.
d*x*	Set the delivery mode to *x*. Delivery modes are ''i'' for interactive (synchronous) delivery, ''b'' for background (asynchronous) delivery, and ''q'' for queue only – that is, actual delivery is made the next time the queue is run.
D	Try to automatically rebuild the alias database, if necessary.
e*x*	Set error processing to mode *x*. Valid modes are ''m'' to mail back the error message, ''w'' to ''write'' back the error message (or mail it back if the sender is not logged in), ''p'' to print the errors on the terminal (default), ''q'' to throw away error messages (only exit status is returned), and ''e'' to do special processing for the BerkNet. If the text of the message is not mailed back by modes ''m'' or ''w'' and if the sender is local (this machine), a copy of the message is appended to the file ''dead.letter'' in the sender's home directory.
F*mode*	Cause **sendmail** to use this mode when creating temporary files. (See **chmod**(1)).
f	Save UNIX From lines at the front of messages.
g*N*	Use *N* as the default group ID when calling mailers.
H*file*	Call the SMTP help file.
i	Ignore a dot on a line by itself as a message terminator.
L*n*	Set the log level.
m	Send to ''me'' (the sender) also, if I am in an alias expansion.
o	Use old-style headers, if possible. If not set, this message is guaranteed to have new-style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
Q*queuedir*	Select the directory in which to queue messages.
r*timeout*	Set the timeout on reads. If none is set, **sendmail** will wait indefinitely for a mailer. This option violates the word (if not the intent) of the SMTP specification; show the timeout should probably be fairly large.
S*file*	Save statistics in the named file.

s Restart the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.

T*time* Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (for example, because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is three days.

t*stz,dtz* Set the name of the time zone. *Stz* is standard time zone; *dtz* is daylight time zone.

u*N* Set the default user ID for mailers to *N*.

sendmail returns an exit status describing what it did. The codes are defined in *<sysexits.h>*

EX_OK	Successful completion on all addresses.
EX_NOUSER	Username not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as **newaliases**, **sendmail** will rebuild the alias database. If invoked as **mailq**, **sendmail** will print the contents of the mail queue.

EXAMPLES

The following alias pipes **sendmail** to the **msgs**(1) command:

msgs: "|/usr/ucb/msgs -s"

An alias such as:

poets: ":include:/usr/local/lib/poets.list"

would read **/usr/local/lib/poets.list** for the list of addresses making up the group.

NOTES

If you are running **sendmail** as a daemon in BSD, you must run **newaliases** on the same node that is running the **sendmail** daemon. Once **newaliases** has completed, you must kill and restart the **sendmail** daemon for the changes to take effect.

FILES

Except for **/usr/lib/sendmail.cf**, these pathnames are all specified in **/usr/lib/sendmail.cf**. Thus, these values are only approximations.

/usr/lib/aliases	raw data for alias names, in text
/usr/lib/aliases.pag	database of alias names used by **sendmail**
/usr/lib/aliases.dir	database of alias names used by **sendmail**

/usr/lib/sendmail.cf	configuration file
/usr/lib/uucpproto.cf	example uucp configuration file
/usr/lib/arpaproto.cf	example ARPANET configuration file
/usr/lib/sendmail.st	collected statistics
/usr/bin/uux	to deliver **uucp** mail
/usr/spool/mqueue/*	temp files
/usr/lib/sendmail.hf	help files
/usr/lib/sendmail.fc	frozen configuration

SEE ALSO

binmail(1), mail(1), rmail(1) syslog(3), aliases(5), sendmail.cf(5), mailaddr(7), rc(8);
DARPA Internet Request For Comments RFC819, RFC821, RFC822;
Using Your BSD Environment.

NAME

server – run a server process

SYNOPSIS

/etc/server [–p] "*command-pathname arg1 arg2 ...*"

DESCRIPTION

The **server** command runs a program with an identity of user "user" and group "server" just as if the command had been started using the Display Manager's **cps** command. It also marks the new process in such a way that the server program will not be terminated by the Display Manager when the user logs out.

In addition to allowing users to start server processes, this command is useful for killing servers that are running with the identity **user.server**. For example,

$ **/etc/server /bin/kill –9 1532**

OPTIONS

–p The **–p** option preserves the current SID of the person issuing the command. Otherwise, **/etc/server** sets the SID to **user.server.none** for the command.

SEE ALSO

init(8), rc(8)

NAME

show_lc – shell script to indicate obsoleted system calls in a binary file

SYNOPSIS

/etc/show_lc *binary_file* ...

DESCRIPTION

This script will show which calls in an object module have been obsoleted and replaced. This will not show which procedure made the call. Generate an **xref** listing to do that. Nor will it show any use of the old **name_$name_t** or **name_$pname_t** data types. You must ensure that the object being examined is in fact an object module.

The *binary_file* argument is required.

SEE ALSO

nm(1)

NAME

shutdown – close down the system at a given time

SYNOPSIS

/etc/shutdown [**–k**] [**–r**] [**–h**] [**–n**] *time* [*warning-message* ...]

DESCRIPTION

shutdown provides an automated shutdown procedure which a super-user can use to notify users nicely when the system is shutting down, saving them from system administrators, hackers, and gurus, who would otherwise not bother with niceties.

time is the time at which **shutdown** will bring the system down and may be the word **now** (indicating an immediate shutdown) or specify a future time in one of two formats: +number and hour:min. The first form brings the system down in *number* minutes and the second brings the system down at the time of day indicated (as a 24–hour clock).

At intervals which get closer together as the shutdown approaches, warning messages are displayed at the terminals of all users on the system. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating **/etc/nologin** and writing a message there. If this file exists when a user attempts to log in, **login**(1) prints its contents and exits. The file is removed just before **shutdown** exits.

At shutdown time a message is written in the system log, containing the time of shutdown, who ran shutdown and the reason. Then a terminate signal is sent to **init** to bring the system down to single-user state.

The time of the shutdown and the warning message are placed in **/etc/nologin** and should be used to inform the users about when the system will be back up and why it is going down (or anything else).

OPTIONS

–r Exec **reboot**(8).

–h Exec **halt**(8),

–k Avoid shutting the system down. (**–k** is to make people think the system is going down!)

–n Prevent the normal **sync**(2) before stopping.

The **–f** option is not supported by Domain/OS BSD.

FILES

/etc/nologin	tells login not to let anyone log in

CAUTIONS

Only allows you to kill the system between now and 23:59 if you use the absolute time for shutdown.

SEE ALSO

login(1), reboot(8)

NAME

sync – update the super block

SYNOPSIS

/etc/sync

DESCRIPTION

sync executes the **sync** system primitive. **sync** can be called to ensure that all disk writes have been completed before the processor is halted in a way not suitably done by **reboot**(8) or **halt**(8). Generally, it is preferable to use **reboot** or **halt** to shut down the system, as they may perform additional actions such as resynchronizing the hardware clock and flushing internal caches before performing a final **sync**.

See **sync**(2) for details on the system primitive.

SEE ALSO

sync(2), fsync(2), halt(8), reboot(8), update(8)

NAME

syncids – fix or verify file owners in a file system

SYNOPSIS

/etc/syncids [**–a**] [**–l**] [**–v**] *volume-pathname*

DESCRIPTION

In a Domain system, every user (and group) is identified by a 64-bit identifier that is unique across all Domain users (and groups) in both space and time. However, UNIX interfaces that take user and group IDs as arguments expect integers that may be unique only within a given file system. The Domain file system stores both forms of identifier with each file. The 64-bit UIDs are used for checking access rights, since there can never be conflicts even if two networks are merged, or a workstation moves from one network to another.

The integer UNIX IDs are used avoid performing expensive translations for operations such as **stat**(2) and **chown**(2).

syncids is a program that should be run whenever network registries are merged, or a node is moved between networks having different registries. It ensures that the UNIX owner IDs match the unique owner IDs for every object on the logical volume.

OPTIONS

–a List the name and owner information for each object as it is processed.

–l Only list information about objects for which UNIX owner IDs are incorrect.

–v Verify only; do not actually modify the owners of any objects. This option implies **–l**. If the **–a** option is also given, then information will be printed about every object on the volume.

SEE ALSO

stat(2),
fstat(2),
chown(2),
chown(8),
edrgy(8).

NAME

syslogd – log systems messages

SYNOPSIS

/etc/syslogd [**–f***configfile*] [**–m***markinterval*] [**–d**]

DESCRIPTION

syslogd reads and logs messages into a set of files described by the configuration file **/etc/syslog.conf.** Each message is one line. A message can contain a priority code, marked by a number in angle braces at the beginning of the line. Priorities are defined in **<sys/syslog.h>**. **syslogd** reads from the UNIX domain socket **/dev/log**, from an Internet domain socket specified in **/etc/services**.

syslogd configures when it starts up and whenever it receives a hangup signal. Lines in the configuration file have a *selector* to determine the message priorities to which the line applies and an *action*. The *action* field is separated from the selector by one or more tabs.

Selectors are semicolon separated lists of priority specifiers. Each priority has a *facility* describing the part of the system that generated the message, a dot, and a *level* indicating the severity of the message. Symbolic names may be used. An asterisk selects all facilities. All messages of the specified level or higher (greater severity) are selected. More than one facility may be selected using commas to separate them.

For example:

***.emerg;mail,daemon.crit**

Selects all facilities at the **emerg** level and the **mail** and **daemon** facilities at the **crit** level.

Known facilities and levels recognized by **syslogd** are those listed in **syslog**(3) without the leading "LOG_". The additional facility **mark** has a message at priority LOG_INFO sent to it every 20 minutes (this may be changed with the **–m** flag). The **mark** facility is not enabled by a facility field containing an asterisk. The level **none** may be used to disable a particular facility.

For example,

***.debug;mail.none**

Sends all messages *except* mail messages to the selected file.

The second part of each line describes where the message is to be logged if this line is selected. There are four forms:

- A filename (beginning with a leading slash). The file will be opened in append mode.
- A hostname preceeded by an at sign (@). Selected messages are forwarded to the **syslogd** on the named host.
- A comma separated list of users. Selected messages are written to those users if they are logged in.
- An asterisk. Selected messages are written to all logged-in users.

Blank lines and lines beginning with '#' are ignored.

syslogd creates the file **/etc/syslog.pid**, if possible, containing a single line with its process id. This can be used to kill or reconfigure **syslogd.**

To bring **syslogd** down, it should be sent a terminate signal (for example, **kill `cat /etc/syslog.pid`**).

OPTIONS

The flags are:

–f	Specify an alternate configuration file.
–m	Select the number of minutes between mark messages.
–d	Turn on debugging.

EXAMPLE

The configuration file:

```
kern,mark.debug          /dev/console
*.notice;mail.info       /usr/spool/adm/syslog
*.crit                   /usr/adm/critical
kern.err                 @ucbarpa
*.emerg                  *
*.alert                  eric,kridle
*.alert;auth.warning     ralph
```

logs all kernel messages and 20 minute marks onto the system console, all notice (or higher) level messages and all mail system messages except debug messages into the file **/usr/spool/adm/syslog**, and all critical messages into **/usr/adm/critical**; kernel messages of error severity or higher are forwarded to **ucbarpa**.

All users will be informed of any emergency messages, the users **eric** and **kridle** will be informed of any alert messages, and the user **ralph** will be informed of any alert message, or any warning message (or higher) from the authorization system.

FILES

/etc/syslog.conf	the configuration file
/etc/syslog.pid	the process id
/dev/log	Name of the UNIX domain datagram log socket

SEE ALSO

logger(1)

NAME

talkd – remote user communication server

SYNOPSIS

/etc/talkd

DESCRIPTION

talkd is the server that notifies a user that somebody else wants to initiate a conversation. It acts a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation. In normal operation, a client, the caller, initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP (see **<protocols/talkd.h>**). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee's login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

SEE ALSO

talk(1), write(1)

NAME

tcpd – TCP/IP protocol server

SYNOPSIS

/etc/tcpd [**–a**] [**–d**<*mask*>] [**–p**<*time*>] [**–t**<*ipttl*>] [**–w**<*window*>]

DESCRIPTION

Invoking **tcpd**, the TCP/IP protocol server or daemon, enables a node's TCP/IP socket-call interface and initializes several internal tables required for operation of the protocols. The **tcpd** daemon must be run on every node that uses TCP/IP. Normally, **tcpd** is invoked by the node's startup file, **/etc/rc**.

Options available with this command allow you to define certain parameters of the protocols.

OPTIONS

–a Suppress delayed packet acknowledgements (ACKs), the default condition. Delayed ACKs is a performance optimization feature for TCP which allows a receiver to delay until 33% of the maximum window size offered can be uncovered (but in no case to wait longer than 0.25 seconds) before it acknowledges received packets.

–p<*time*> Set gateway ping interval in seconds. The value must be expressed in hexadecimal and can range from 0 to A. Default value is 4 seconds. Setting the interval to 0 inhibits pinging. TCP issues an ICMP Echo Request (ping) to local hosts and gateways at the specified interval to verify their continued operation. Any gateway that fails to respond after three tries is moved to the end of the node's routing table.

–t<*ipttl*> Set the IP parameter, packet maximum time to live. The value must be expressed in hexadecimal and can range from 1 to FF. Default value is 1E.

–w<*window*> Set the receive window size (a TCP flow control parameter) in octets. The value must be expressed in HEX and can range from 1 to 239C. Default value is 239C.

–d<*mask*> Display debugging information as defined by the 16-bit mask. See the table below for a description of the debug information that can be requested. Add bit values to request several types of information.

{_Bit Value_}	{_Debug Information_}
0001 (default)	General information
0002	IP level information
0004	ARP information
0008	TCP information
0010	Data in TCP packets
0020	UDP information
0200	Broadcasts
1000	TCP finite state machine information
2000	Device level information
4000	Additional detail at any level
foff	All available debug information except broadcasts

SEE ALSO

netstat(1), ping(8);
Configuring and Managing TCP/IP.

NAME

telnetd – DARPA TELNET protocol server

SYNOPSIS

/etc/telnetd

DESCRIPTION

telnetd is a server which supports the DARPA standard **TELNET** virtual terminal protocol. **telnetd** is invoked by the internet server (see **inetd**(8)), normally for requests to connect to the **TELNET** port as indicated by the **/etc/services** file (see **services**(5)).

telnetd operates by allocating a pseudoterminal device (see **pty**(4)) for a client, then creating a login process which has the slave side of the pseudoterminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the pseudoterminal, implementing the **TELNET** protocol and passing characters between the remote client and the login process.

When a **TELNET** session is started up, **telnetd** sends **TELNET** options to the client side indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal type information* from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudoterminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see **tty**(4)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **telnetd** is willing to have the remote client *do*: *binary*, *terminal type*, and *suppress go ahead*.

BUGS

Some **TELNET** commands are only partially implemented.

The **TELNET** protocol allows for the exchange of the number of lines and columns on the user's terminal, but **telnetd** doesn't make use of them.

Because of bugs in the original 4.2BSD **telnet**, **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2BSD **telnet**.

Binary mode has no common interpretation except between similar operating systems (the UNIX system in this case).

The terminal type name received from the remote client is converted to lowercase.

The *packet* interface to the pseudoterminal (see **pty**(4)) should be used for more intelligent flushing of input and output queues.

telnetd never sends **TELNET** *go ahead* commands.

SEE ALSO

telnet(1C)

NAME

tftpd – tftp daemon

SYNOPSIS

/etc/tftpd

DESCRIPTION

tftpd is a daemon which runs the trivial file transfer protocol server for the BSD Internet software. It is called by **inetd**(8C) when an incoming datagram requests the **tftp**(1C) service. It then handles **tftp**(1C) file transfers in accordance with RFC783.

Note that **/etc/tftpd** must be setuid to a designated **tftp**(1C) user (usually a very non-privileged userid and never root) and that the **tftp**(1C) spool directory must be owned by that user.

NOTES

The Domain/OS BSD versions of **tftp**(1C) and **tftpd** are adaptations of the Massachusetts Institute of Technology implementations of the **tftp** protocol. Domain/OS BSD **tftp** will interface with any RFC783 compliant implementation. Note, however, that the 4.3BSD distribution version of **tftp** does not meet these restrictions.

WARNINGS

tftp(1C) and **tftpd** comprise an implementation of the Trivial File Transfer Protocol described in RFC783. They allow you to quickly copy files among hosts on an internet without regard to ownership or access restrictions. Therefore, the desired security of a system should be considered before allowing **tftp**(1C) transactions. In an inspired attempt to prevent accidental destruction of important files, **tftp**(1C) requires that remote file names be absolute pathnames (that is, beginning with /) containing the string "**/tftp/**", but not containing the string "/../".

SEE ALSO

tftp(1C), inetd(8C);
Configuring and Managing TCP/IP.

NAME

trpt – transliterate protocol trace

SYNOPSIS

trpt [**–a**] [**–c**] [**–d** <*PCB addr*>] [**–e**] [**–f**] [**–j**] [**–l**] [**–s**] [**–t**] [**–w**] [**–p** <*PCB addr*>] [<*filename*>]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **setsockopt**(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system, grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

–a	Print the values of the source and destination addresses for each packet recorded, in addition to the normal output.
–f	Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.
–j	Just give a list of the protocol control block addresses for which there are trace records.
–p <*PCB addr*>	Show only trace records associated with the protocol control block, the address of which follows.
–s	Print a detailed description of the packet sequencing information, in addition to the normal output. (Currently unimplemented)
–t	Print the values for all timers at each point in the trace, in addition to the normal output. (Currently unimplemented)

Domain/OS BSD EXTENSIONS

–c	Clear trace buffer.
–d <*PCB addr*>	Toggle debug on a connection.
–e	Exit on a bad trace record.
–l	Print lapsed times, in addition to the normal output.
–w	Warn on bad trace records.

NOTES

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the –A option to **netstat**(1). Then run **trpt** with the **–p** option, supplying the associated protocol control block addresses. The **–f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a file other than the default, which is **'node_data/systmp/tcp_data**, <*filename*> may be used to specify another file.

DIAGNOSTICS

```
no namelist
```

Printed when the system image doesn't contain the proper symbols to find the trace buffer.

Other diagnostics should be self explanatory.

CAUTIONS

trpt should print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable.

FILES

'node_data/systmp/tcp_data

SEE ALSO

setsockopt(2), netstat(1);
Configuring and Managing TCP/IP.

NAME

uctnode – uncatalog a node

SYNOPSIS

/etc/uctnode [*options*] *pathname* ...

DESCRIPTION

uctnode removes the specified entry directory name from the local copy of the network root directory. After the name is removed, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

If you use the **–root** option, the nodename is also removed from the network's replicated root directory.

Node entry directories are created with the **ctnode** command.

pathname (required) Specify node entry directory name to be uncataloged. Multiple pathnames and wildcarding are permitted.

OPTIONS

–l List directory names as they are uncataloged.

–root Uncatalog the node in the network root as well as in the the local root directory.

EXAMPLES

Uncatalog the node with the entry directory name specified.

```
$ /etc/uctnode als_node
```

NAME

uctob –uncatalog the specified pathname, without deleting the associated object.

SYNOPSIS

/etc/uctob [**–br**] *pathname...*

DESCRIPTION

The command **uctob** removes the specified pathname from the name space. The object associated with the pathname is not affected. This command is primarily intended for system-level debugging use.

OPTIONS

–br Suppress listing of names and uids of objects as they are uncataloged. These are reported unless this option is specified.

EXAMPLES

```
$ /etc/uctob testfile
"testfile" uid is 16791C0C.40000074.
```

This example uncatalogs **testfile**.

NAME

ulkob – unlock an object

SYNOPSIS

/etc/ulkob [*options*] [*pathname* ...]

DESCRIPTION

ulkob unlocks objects residing on, or locked by processes running on, the current node. You cannot unlock objects on remote nodes unless you locked them (see **–f** below).

pathname (optional)

Specify name of object to be unlocked. Multiple pathnames and wildcarding are permitted.

Default if omitted: **–u** option must be specified

OPTIONS

If no options are specified, the object is unlocked for all lock modes.

–r Unlock an object that was locked for read mode; the lock must be owned by this process.

–w Unlock an object that was locked for write mode; the lock must be owned by this process.

–i Unlock an object that was locked for reading with intent to write; the lock must be owned by this process.

–f Forcibly unlock an object. It may have been locked for any mode and the lock may be owned by any process. The object must reside on the current node, however, or must have been locked by the current node. In other words, you cannot unlock objects on a remote node unless you locked them.

–l List the name of each object as it is unlocked.

–u *uid* ... Specify the UID of the object(s) to unlock. Multiple UIDs are permitted. If the *pathname* argument is omitted, then this option is required.

EXAMPLES

Forcibly unlock the file **mary** for any mode and unlock the two objects with the specified UIDs.

```
$ /etc/ulkob mary –f

$ /etc/ulkob –uid 1C1A9E2F.20000246 1C1A9E42.50000246
```

NAME

update – periodically update the super block

SYNOPSIS

/etc/**update**

DESCRIPTION

update is a program that executes the **sync**(2) primitive every 30 seconds. This ensures that the file system is fairly up to date in case of a crash. This command should not be executed directly, but should be executed out of the initialization shell command file.

SEE ALSO

sync(2), sync(8), init(8), rc(8)

NAME

uucheck – check the **uucp** directories and permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x** *debug_level*]

DESCRIPTION

uucheck checks for the presence of the **uucp** system required files and directories. Within the **uucp** makefile, it is executed before the installation takes place. It also checks for some obvious errors in the **Permissions** file (**/usr/lib/uucp/Permissions**).

OPTIONS

–v Gives a detailed explanation of how the **uucp** programs interpret the **Permissions** file.

–x *debug_level*

Used for debugging. *debug_level* is a single digit in the range 1-9; the higher the value, the greater the detail.

NOTE

uucheck can only be used by the super-user or **uucp**.

BUGS

The program does not check file/directory modes or some errors in the **Permissions** file, such as duplicate login or machine name.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucico(8C), uusched(8C);
uucp(1C) uustat(1C) uux(1C) in the BSD Command Reference.

NAME

uucico – file transport program for the **uucp** system

SYNOPSIS

/usr/lib/uucp/uucico [**–r** *role_number*] [**–x** *debug_level*] [**–i** *interface*] [**–d** *spool_directory*] **–s** *system_name*

DESCRIPTION

uucico is the file transport program for **uucp** work file transfers. Domain/OS systems supply two versions of **uucico**: **uucico**, which, when called simply exits, leaving your work in the queue, and **uucico.real**, which, as its name implies, is the "real" **uucico** program. Normally, **uucico.real** is invoked locally by **cron**, or when a remote host initiates a **uucp** connection with the Domain/OS system in slave mode. Work queued by you is processed when **uucico.real** is invoked by either method. The only way to directly invoke **uucico.real** is to run it on the node on which it is installed.

OPTIONS

–r*role_number*

role_numbers for the master mode and slave mode. One for master mode or zero for slave mode (default). **–r** should be specified as the digit 1 for master mode when **uucico** is started by a program or **cron**. **uux** and **uucp** both queue jobs that are to be transferred by **uucico**. It is normally started by the scheduler, **uusched**, but can be started manually; this is done for debugging. For example, the shell **Uutry** starts **uucico** with debugging turned on.

–x*debug_level*

You must use a single digit for this option, with higher numbers for more debugging.

–i*interface* Defines the interface used with **uucico**. This interface only affects slave mode. Known interfaces are the UNIX System (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write).

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Devconfig
/usr/lib/uucp/Sysfiles
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(8), uusched(8C), Uutry(8C),
uucp(1C), uustat(1C), uux(1C), in the *BSD Command Reference*.

NAME

uuclean – **uucp** spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uuclean *options*

DESCRIPTION

uuclean scans the spool directory for files with the specified prefix (see OPTIONS below) and deletes all files that are older than a specified number of hours.

This program is typically started by **cron**(8).

OPTIONS

–d*directory* Cleans *directory* instead of the spool directory. (**Note**: if *directory* is not a valid spool directory, it cannot contain "work files", that is, files whose names start with "C". These files have special meaning to **uuclean** pertaining to **uucp** job statistics.)

–p*pre* Scans for files with *pre* as the file prefix. You can specify up to 10 **–p** arguments. Using **–p** without *pre* following causes all files older than the specified time to be deleted.

–n*time* Deletes files whose age is more than *time* hours if the prefix test is satisfied (the default time is 72 hours).

–w*file* Finds files older than *time* hours, but does not delete them. If the *file* argument is present, a warning is placed in *file*. Otherwise, warnings go to the standard output.

–s*sys* Examines only those files destined for system *sys*. You can specify up to 10 **–s** arguments.

–m*file* Sends mail to the owner of the file when it is deleted. If a *file* is specified, a warning is also placed in *file*.

FILES

/usr/lib/uucp directory with commands used by **uuclean** internally
/usr/spool/uucp spool directory

SEE ALSO

cron (8)
uucp (1C), in the *BSD Command Reference*.

NAME

uucleanup – **uucp** spool-directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [–C*time*] [–W*time*] [–X*time*] [–m*string*] [–o*time*] [–s*system*]

DESCRIPTION

uucleanup scans the spool directories for old files and takes appropriate action to remove them in a useful way. **uucleanup** informs you of send/receive requests for systems that cannot be reached. It returns mail that cannot be delivered. It deletes or executes **rnews** for **rnews** type files (depending on whether the news originated locally or remotely). **uucleanup** removes all other files.

In addition, you are warned about requests that have been waiting for a given number of days (the default is 1.)

OPTIONS

–C*time* Removes any **C.** files greater or equal to *time* days old and supplies you with the appropriate information. (the default is 7 days.)

–D*time* Removes any **D.** files greater or equal to *time* days old. Attempts to deliver mail messages and execute **rnews** when appropriate. (the default is 7 days.)

–W*time* Sends a mail messsage concerning any **C.** files equal to *time* days old warning about the delay in contacting the remote. The message includes the JOBID, and in the case of mail, the mail message. The administrator can include a message line telling who to call to check the problem (see **–m** option). (the default is 1 day)

–X*time* Removes any **X.** files greater or equal to *time* days old. **D.** files are probably not present (if they were, **X.** should have been executed). But if there are **D.** files, they are taken care of by **D.** processing. (the default is 2 days.)

–m*string* Includes this line in the warning message generated by the **–W** option.

–o*time* Deletes other files older than *time* days old. (the default is 2 days.) The default line is "See your local administrator to locate the problem".

–s*system* Executes for *system* spool directory only.

–x*debug_level*

debug_level is a single digit between 0 and 9; higher numbers give more detailed debugging information. (If **uucleanup** is compiled with -DSMALL, no debugging output is available.) This program is typically started by the shell **uudemon.cleanup**, which should be started by **cron**(8C).

NOTE

uucleanup works as if all option *times* are specified to the default values unless you specifically set *time*.

FILES

/usr/lib/uucp

Directory with commands used by **uucleanup** internally **/usr/spool/uucp** Spool directory

SEE ALSO

cron (8C), uucp(1C), uux(1C)

NAME

uuid_gen – UUID generating program

SYNOPSIS

/etc/ncs/uuid_gen [–c] [–p] [–C] [–P]

DESCRIPTION

The **uuid_gen** program generates Universal Unique Identifiers (UUIDs). Without options, it generates a character-string representation of a UUID. The options enable you to generate templates for Network Interface Definition Language (NIDL) files or to generate source-code representations of UUIDs, suitable for initializing variables of type **uuid_$t**.

OPTIONS

–c	Generate a template, including a UUID attribute, for an interface definition in the C syntax of NIDL.
–p	Generate a template, including a UUID attribute, for an interface definition in the Pascal syntax of NIDL.
–C	Generate a C source-code representation of a UUID.
–P	Generate a Pascal source-code representation of a UUID.

EXAMPLES

Generate a character-string representation of a UUID:

```
$ /sys/ncs/uuid_gen
34dc23469000.0d.00.00.7c.5f.00.00.00
```

Generate a template for an interface definition in the C syntax of NIDL:

```
$ /sys/ncs/uuid_gen –c
%c
[
uuid(34dc239ec000.0d.00.00.7c.5f.00.00.00),
version(1)
]
interface INTERFACENAME {
```

Generate a C source-code representation of a UUID:

```
$ /sys/ncs/uuid_gen –C
= { 0x34dc23af,
0xf000,
0x0000,
0x0d,
{0x00, 0x00, 0x7c, 0x5f, 0x00, 0x00, 0x00} };
```

NAME

uusched – the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusched [–x*debug_level*] [–u*debug_level*]

DESCRIPTION

uusched is the **uucp** file transport scheduler. It is usually started by the daemon **uudemon.hour** that is started by **cron**(8) from an entry in **/usr/spool/cron/crontab**:

```
39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"
```

OPTIONS

–x*debug_level*	Causes debugging messages to be output from *uusched*.
–u*debug_level*	Causes *debug_level* to pass as –x *debug_level* to **uucico**. *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

NOTE

The two options are for debugging purposes only.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(8), uucico(8C),
uucp(1C), uustat(1C), uux(1C), in the *BSD Command Reference*.

NAME

uuxqt – execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [–s*system*] [–x*debug_level*]

DESCRIPTION

uuxqt executes remote job requests from remote systems generated by the use of the **uux** command. (**Mail** uses **uux** for remote mail requests). **uuxqt** searches the spool directories looking for *X.* files. For each *X.* file, **uuxqt** checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. **uuxqt** uses the *Permissions* file to validate file accessibility and command execution permission.

Two environment variables are set before **uuxqt** is executed:

UU_MACHINE is the machine that sent the job (the previous one).
UU_USER is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

debug_level is a single digit between 0 and 9. Higher numbers give more detailed debugging information.

FILES

/usr/lib/uucp/Permissions
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*

SEE ALSO

uucico(8C)
uucp(1C), uustat(1C), uux(1C), mail(1), in the *BSD Command Reference*.

NAME

ver – change the version of shell commands

SYNOPSIS

ver
ver *systype*
ver *systype command*

DESCRIPTION

ver changes, temporarily or permanently, the UNIX version of commands that are executed by the shell. The command also displays the version in use.

The first form of the command (without arguments) displays the current value of the **systype** variable, which specifies the UNIX version of commands being executed by the shell. The second form allows you to set the UNIX version to the specified *systype*. Valid *systype* values are **sys5.3** and **bsd4.3**.

The third form of the command causes the UNIX version of *command* specified by **systype** to be executed, with no permanent change to **systype**'s value.

FILES

/etc/ver	used for the third form of the command

SEE ALSO

systype(1);

NAME

writed – daemon for **write**(1) program

SYNOPSIS

/etc/writed

DESCRIPTION

The **writed** daemon is used by the **write**(1) program. To enable a **write** to a node other than your own, **writed** must be running on the destination node. Similarly, you must be running **writed** on your node if any node is to be able to write back to you.

We recommend invoking **writed** in background mode via the **/etc/rc.user** shell script. If you include the lines

```
if [ -f /etc/writed ]; then
       /etc/writed &
fi
```

in your node's `**node_data/etc/rc** file, the **init** process will start **writed** at boot time (after first checking to see if the file exists). We include the above lines on installation of each node, except for those that are diskless.

The **writed** process starts an **mbx_helper** automatically when it is invoked.

FILES

/etc/utmp	record of who is logged in on the node (link to **node_data/etc/utmp**)

SEE ALSO

write(1), utmp(5), rc(8)

Appendix A

Using the netmain Interactive Tool and netmain_srvr in the Apollo Token Ring Network

Contents

Appendix A

Using the netmain Interactive Tool and netmain_srvr in the Apollo Token Ring Network

This appendix provides guidelines for using network topology lists and data gathered by **netmain_srvr** and the **netmain** interactive tool displays to draw conclusions about your Apollo Token Ring network's performance. It contains methods to detect conditions that are potentially troublesome, and thus is oriented toward problem prevention. This appendix also contains information on routine maintenance to protect the physical integrity of your network. It includes information about

- Node performance
- Detecting intermittent network events
- Problem isolation methods
- Routine maintenance procedures

A.1 Solving Network Problems with netmain and netmain_srvr

This section provides step-by-step procedures that can help you to start using the **netmain** menus. These procedures show you how to move through the menus and perform some of the basic operations needed to control **netmain_srvr**. Refer to later sections for a full description of the **netmain** interactive tool menus and the **netmain_srvr** options.

A.1.1 Starting and Stopping the netmain_srvr

To start the **netmain_srvr**, use one of the methods described below.

Starting netmain_srvr from the DM Command Line

To start the **netmain_srvr** from the DM command line, type:

Command: **/etc/server netmain_srvr** [*-options*]

The server process begins immediately and continues after logout.

Starting the netmain_srvr from a Start-Up File

To automatically start **netmain_srvr** server, uncomment the following lines in the disked node's **/etc/rc.user** file.

```
#if [ -f  /sys/net/netmain_srvr  ]; then
#  (echo " netmain_srvr\c" >/dev/console)
#  /sys/net/netmain_srvr &
#fi
```

The server process begins when the node comes online and continues after logout. If you don't wish to use the entire set of defaults, you can specify those you wish to use with options in the **netmain_srvr** command line or configuration file. See Section A.4 for details on options.

Stopping the netmain_srvr

When started by the methods described above, **netmain_srvr** continues running until it is intentionally stopped with either of the following commands:

% kill netmain_srvr -q (UNIX operating system environments)

% sigp netmain_srvr -q (Aegis operating system environment)

The server process stops running if the node is shut down intentionally or if the system crashes. If the process stops running, you may start it from the DM command line or by rebooting the node.

A.1.2 Getting Started with netmain_srvr and netmain

Before you can practice using **netmain**, a monitor has to run in your network. If no monitors are running in your network, start **netmain_srvr** as described above. Let the monitor run, with default values, for a week. Then you may begin Procedure A-2. If you already have had **netmain_srvr** running for more than a week, proceed directly to Procedure A-2.

Procedure A–2 *Getting Started with Netmain*

Task 1: Invoke netmain.

To invoke **netmain**, type:

```
% netmain -whelp
```

The **-whelp** option tells **netmain** to grow the window, to display the entire **netmain** menu, including the help area. The Top–Level menu, the top of which is shown below, appears.

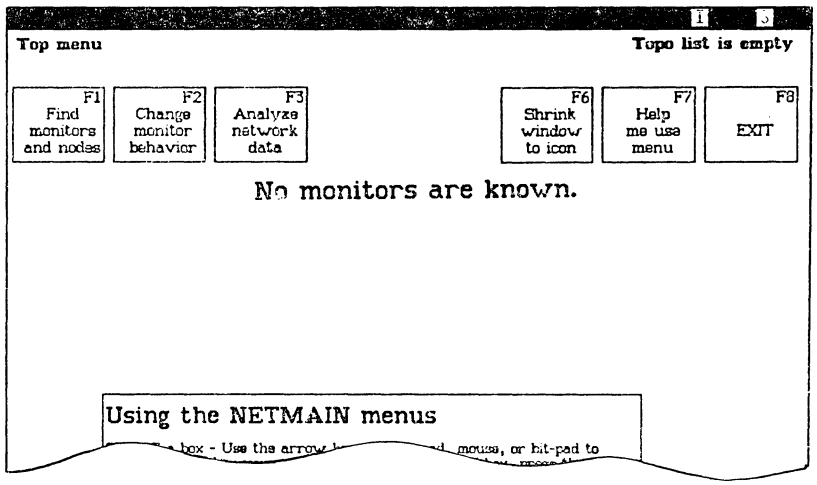


Figure A–1. Top–Level Menu

Task 2: Study the menu.

The **menu name** appears in the top–left corner.

Each **command box**, across the top of the menu, shows a command and the keyboard **function keys**, F1 through F8.

Netmain messages that describe functions selected appear below the command boxes. The message, "No monitors are known," below the command boxes reminds you that you haven't yet chosen a monitor. **Netmain** also uses this area to draw an input box and to prompt you for input.

Error and status messages appear above the command boxes. **Netmain** reports on its activity as it carries out commands. There are no messages above the command box right now. A "Topology List status" message appears in the top right corner.

The large box at the bottom of the screen contains online help about each menu.

Task 3: Locate the + shaped cursor in the netmain window.

Use the arrow keys, touch–pad, mouse, or bitpad to move the cursor. Move the cursor outside the window. It regains its rectangular shape and the Display Manager is in control. Move the cursor back to the **netmain** window.

Taks 4: Point to a command box by placing the cursor in the box.

Notice that pointing to a command box accents the box. It changes color. When a box is accented, the + shaped cursor is no longer visible.

Task 5: Point to the "F6 Shrink window to icon" command box to make the netmain window into an icon.

To get help on any **netmain** command or parameter, type **h**, or **?**, or press the middle or right buttons on the mouse, or press any button except the top on the bit–pad puck. The menu help box displays additional information about the command.

Task 6: Select the "F6 Shrink window to icon" command.

To select a box in a **netmain** menu, you can use one of the following methods:

- Press the appropriate function key (only for command boxes labeled with function key names).
- Point to the box and press the keyboard's space bar.
- Point to the box and press the mouse's left button.
- Point to the box and press the top button of the bit–pad puck.

The **netmain** window becomes an icon. The **netmain** program continues to run in the icon, retaining any information it has amassed, but taking up little space on the screen.

Task 7: Point to the netmain icon by moving the DM cursor over the icon.

Task 8: Request help about the icon by pointing to it and typing h or ?, pressing the middle or right button on the mouse, or pressing any button except the top one on the bit–pad puck.

A system help file is displayed.

Task 9: **Enter netmain by pointing to the icon and pressing the space bar. Then select "F2 Change monitor behavior."**

Note the error message that appears. You have not yet selected a monitor with which to work.

Task 10: **Now look for the command box to exit from netmain.**

As you probably guessed, it's the F8 Exit box. On any **netmain** menu, use this command box to exit to the higher-level menu. On this top menu, use it to exit from **netmain**.

A.1.3 Establishing Network Performance Levels

If users perceive poor network performance, use the information **netmain_srvr** gathers to determine the reason. Sometimes a small or intermittent problem in a connector, cable, or ring interface board can cause network or node performance problems. This section describes how to use the **netmain** interactive tool to look for such problems and isolate them to one point in the network, usually a node's IN and OUT cables and connectors. Sometimes poor distribution of network resources can also cause poor network performance. This section describes how to "tune" the network by searching for subtle problems that can affect performance. Follow the suggestions in this section to prevent problems and enhance network performance. Section A.5 describes how to use the **netmain** interactive tool to detect resource distribution problems.

To increase your understanding of the material in this appendix, use the **netmain** interactive tool and perform the operations described as you read the material. Allow ample time to cover the material, and keep your reading and practice sessions short.

Evaluating Node Performance

The **netmain** interactive tool displays can be used to analyze individual node performance as well as overall network performance. This section describes how to use these displays to evaluate the performance of individual nodes in the network. Questions that require an evaluation of node performance include the following:

- What percentage of network resources are provided by individual nodes, and is this distribution appropriate for the network?
- Are nodes' diskless partner assignments affecting the performance of either node?
- Is any node experiencing a high number of disk or memory errors that could indicate hardware problems?

Locating Underused or Overused Nodes

The CPU_TIME and NET_SERVICE probes gather data about CPU usage and network requests. The categories CPU SERVICE and QUEUE SERVICE from the **netmain** interactive tool's Analyze Network Data menu include performance statistics for data gathered by these probes. To locate underused or overused nodes, look at performance statistics in these categories. Also, use the output formats that report on diskless nodes. This section provides guidelines for using these tools.

Look at the network resources that nodes provide by checking the number of diskless partners assigned to each node. Choose the "Diskless partners" output format and select "Partners, by mother." Check the number of diskless partners assigned to the nodes shown. If some nodes have several diskless partners and others have none, consider reassigning some diskless nodes to other partners. If the nodes are servers, dedicated to serving diskless node, the arrangement may be satisfactory.

Next, choose a "Density across network" output format. It shows each node's relative contribution of a resource as a percentage of the total for the network. Start by looking at the "Total file service" and "Total page service" performance statistics. These statistics show the number of file services and paging services each node performs for other nodes. A node provides these services when other nodes need to read, write, open, close, and list its files. Nodes with diskless partners have higher counts of these services than nodes without diskless partners. Experiment by setting the "top of error scale" level in the Parameter menu to different values. In the output displays, look for the following patterns:

- Sharp, high–density, vertical lines underneath certain nodes, extending across all time intervals
- Sharp, high–density, vertical lines that appear only during certain time intervals

A node with a high–density vertical line is heavily consuming the network resource shown by that performance statistic. Check to see whether such heavily used nodes are servers nodes (typically DSPs, such as the DSP90 or DSP160), and/or whether they have diskless partners (use the "Diskless partners" format).

If a heavily used node has no diskless partners and isn't a server node, it may contain a system resource that can be replicated elsewhere. For example, if commonly used sets of files can be replicated on other nodes, the traffic at this node will be lighter. Note, however, that files are good candidates for replication only if there is a mechanism for coordinating file updates. If a node is heavily used only at certain times (a high–density line appears at certain time intervals, then fades), investigate the activity on that node during these times.

If only servers and/or partners of diskless nodes are used heavily, check the relative contribution of each server. Create a data file containing only the names of the servers. Use the format described in Section A.5 for the F5 (Topo list from data file) command in the Find Monitors and Nodes menu. Put the data file in the topology list. Then use the "Density across network" format to look at the performance statistics again.

Look for high–density vertical lines that indicate heavy loads on individual nodes and also look for high–density horizontal bands. High–density bands can give information about times of peak use in the network. Analyze user activities during these peak periods to determine which activities consume system resources. You may be able to change the "mix" of services on the server nodes to create a more efficient pattern of use.

Use rates or incidence formats to analyze nodes' file and paging services performed for other nodes. To find out about a node's performance during times of peak use, display "page or file backlog severity" performance statistics (from the QUEUE_SERVICE category). Backlog severity statistics report the average number of service requests in the file or paging service queues, at times when there are requests in these queues.

These statistics show the nodes that are most heavily used during peak periods. Traffic in local area networks tends to occur in bursts, and distributing resources to share the load during peak periods can help to provide maximum performance of your network. To compare the contributions of individual nodes, use an across–network display format.

The "any backlog" statistic shows all unserviced requests in a service queue. The "average backlog" statistic shows the average number of requests in a queue over time. Note that a high average backlog or backlog severity level is not an error condition. It shows that the node is providing a relatively high percentage of service to the network. Evaluating this condition depends on what you intend the node to provide, and what you intend other nodes to provide.

If an "any backlog" display shows that some nodes have backlogs and others do not, distribute more resources to nodes without backlogs. If some nodes have a higher average queue length than others, investigate the number of pages of memory that node has allotted for paging requests from remote nodes. If a user has used the **netsvc** shell command with the **-p**[*n*] option, remote nodes are limited to only *n* pages of the node's memory, and this could add slightly to the queue length.

For more information when you suspect a node is overused, look at the "Total disk activity" performance statistic from the **DISK** category. Display the data by using the same rates or incidence formats that you used with the CPU SERVICE and QUEUE SERVICE data. Use the Display Manager (DM) to place one output pad directly over the other. If both displays show a vertical high–density line for the node in question, it is probably overused. To confirm this, compare this node's **DISK** category statistic with those of other nodes in the network. If this node's disk activity is comparable to that of other nodes, it is not, in fact, being overused.

The procedures described above are used to search for nodes that are providing more than their share of network resources. To find nodes that are providing less than their share, use the "Null CPU time" performance statistic in various display formats. Density formats are useful because they allow you to easily compare node's performance.

All nodes have some percentage of Null CPU time. Compare the node's "Null CPU" time to its "Total disk activity" from the **DISK** category, using the "two–display" technique de-

scribed earlier. If the node shows both a high disk activity level and a high Null CPU time level, the node may be spending too much time servicing paging requests and not enough time using the CPU for computation. The node's performance might improve if it were used less as a system resource.

Nodes that show high "Null CPU" time but do not have a high "Total disk activity" level can provide more network services. Nodes without service queue backlogs can also provide more network services. Use a "peaks" format to look for nodes with no service queue backlogs.

Diskless Partner Information

The previous subsection described a quick check to compare diskless partner assignments for nodes of various types. Use the network resource analysis techniques described in this subsection to ensure that nodes are not acting as partners to too many diskless nodes.

To ensure that diskless nodes' partners are in the same network loop, prepare data files with lists of nodes in each network loop. Put the data files into the Topo List (F5 in the Find Monitors and Nodes menu). Then use the "Diskless partners" output format to display diskless nodes and their partner nodes. Verify that each diskless node has a partner in the same loop.

Disk and Memory Errors

Performance statistics in the **DISK** and **STORAGE MODULE** categories, and printed displays on memory errors can alert you to potential hardware problems. In the **DISK** and **STORAGE MODULE** categories, use any graph display format, particularly a peaks format, to check for Disk or SMD (Storage Module Disk) CRC error. Set the "top of error scale" to 5%. If a disk or SMD shows high or increasing levels of CRC errors (above 0.01%), copy user files elsewhere and contact a service representative. Incidences of "Disk/SMD not ready" or "Disk equipment check" conditions also should not occur.

In printed displays of memory errors ,look for sudden occurrences of ECCC or ECCU errors. If these occur, contact a service representative. In the case of ECCC errors, the service representative can determine if there actually is a problem. In the case of ECCU errors, a hardware problem is usually indicated. The service representative may replace a failing board.

A.2 Detecting Unusual or Intermittent Network Events

Use the methods described below to find intermittent or unusual network performance conditions and to recognize these conditions in **netmain** tool displays.

Become familiar with the performance levels typical of your network. Do this by producing plots via the "incidence density output" format. Choose performance statistics in the **RING RECEIVE** or **RING TRANSMIT** categories. Look for horizontal high-density bands ex-

tending across all nodes in the network for a period of time. These high-density bands indicate that the entire network showed increased levels of the performance statistic you are analyzing, over a certain time period.

Compare the times at which horizontal high-density bands appear on plots for different days. If there is any pattern to the time periods, investigate user and network activities occurring during these time periods.

When you start to look for intermittent conditions, check two performance statistics in the RING TRANSMIT category, "Transmit no return" and "Transmit packet error," for levels that rise across the entire network during any kind of a problem.

Select "Transmit no return" and an incidence density format to display the data. Set the "top of error scale" to 25%. If most of the nodes in the network show some gray, there may be a problem worth investigating further. Increased levels of this statistic occur when nodes send a packet but do not get a response in return. High levels for this statistic can indicate a cable or connector problem. Cable or connector problems can cause nodes to send hardware failure messages. Select a "Scattergram events output" format and look at hardware failure messages. Note the nodes that report hardware failure messages. Check the cables and connectors for these nodes and for nodes directly upstream of them.

If there are no hardware failure messages in the scattergram, or if the "Transmit no return" statistics did not indicate problems, look at the "Transmit packet error" performance statistic. Choose an "incidence density" format to display the data. Leave the "top of error percentage scale" at its default value of 50%. If most of the nodes in the network don't show much gray, there is probably not a problem. If most do show gray, choose an "incidence peaks" format and experiment with various "Top of error threshold scale" levels. Note the level at which many nodes start to show peaks of "Transmit packet errors." Also note whether the peaks occur for most nodes or only some nodes.

Isolating a Problem to a Particular Node

This subsection describes how to look at displays of performance statistics to isolate the source of the problems described in the previous subsections.

Choose an "incidence density" format and any of the following performance statistics:

- Receive modem errors (from RING RECEIVE category)
- Receive biphase errors (from RING RECEIVE category)
- Transmit modem errors (from RING TRANSMIT category)
- Transmit biphase errors (from RING TRANSMIT category)

In the displays, look for a vertical, sharp, high-density line under one node. It indicates saturation or near-saturation conditions for that node. If you don't see a high-density line

under any node in the gray scale plot, lower the "top of error scale" and see if the condition occurs. Figure A–2 shows an example of such a condition in a density plot.

If the output shows a high–density vertical line, inspect that node, the node immediately upstream, and all the cables and connections between these nodes. This output can indicate a problem in the node's ring interface hardware that is not affecting other nodes in the network, but sometimes it can point out a problem with cabling or a connection.

Check further by examining output from an active monitor while someone moves the cables in the areas that you suspect. If the count increases, the problem is probably in the physical medium (cables or connectors).

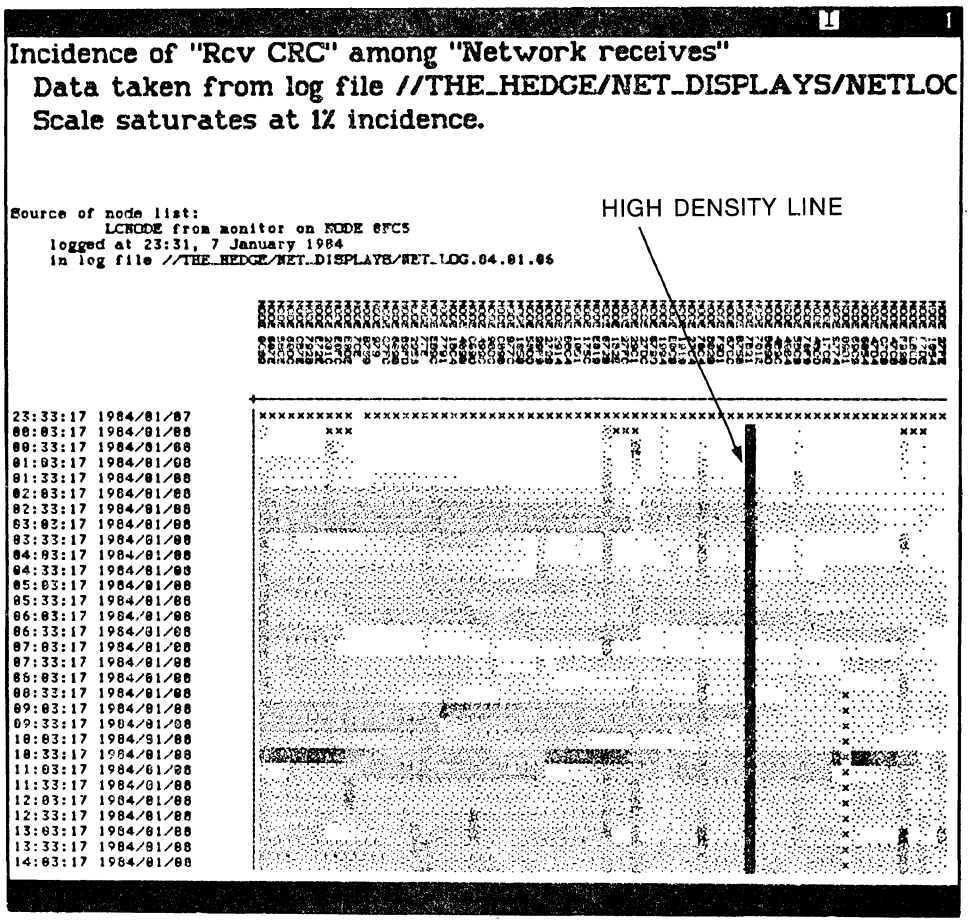


Figure A–2. High–Density Line Condition

If moving the cables does not affect the density, the problem may be in the ring interface hardware for the node that showed the high-density vertical line, or the ring hardware of the node upstream. Use the shell command **netsvc -n** to take one of the nodes off the network while you continue to examine output from an active monitor. If the high-density vertical line starts to disappear, the problem is probably in the node that you removed from the network. While the node is off the network, see if "Transmit packet error" levels go down significantly. If the results show that the node is hindering network performance, schedule maintenance for the node.

Choose an "incidence density" format, and the "Rcv CRC" performance statistic in the RING RECEIVE category. The operating system increments counts for this statistic when part of a received message does not pass the CRC. Thus, this statistic correlates with corruption of one or several bits in a packet.

Set the "Top of error percentage scale" to 1%. Look for areas of high density that fade horizontally into areas of lower density. Figure A-3 shows an example of high-density fade in a gray scale plot.

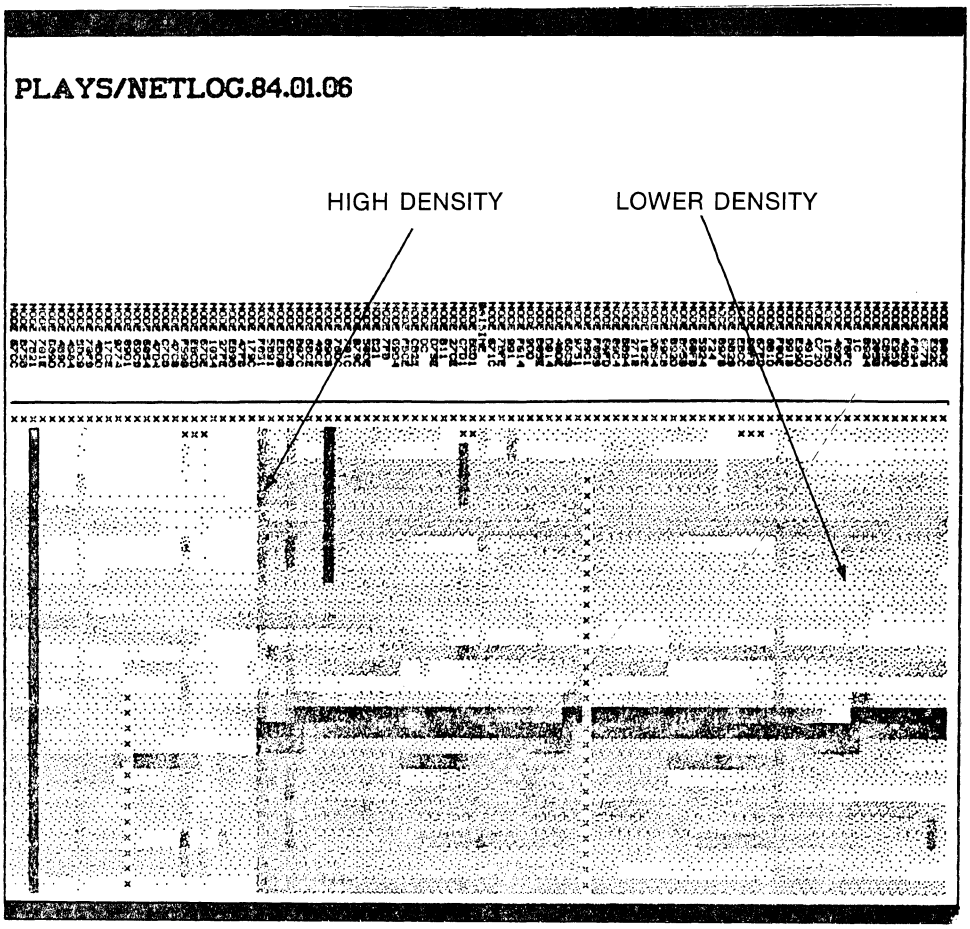


Figure A–3. High–Density Fade Condition

High–density areas that fade into low–density areas indicate a problem in one node that affects data integrity in other nodes. The nodes with the highest density are closest to the problem because almost all the messages they receive must pass through the node causing the problem. Nodes that are further downstream show a lower error density because they receive some messages that haven't passed through the node causing a problem.

If you do not detect the condition, set the "Top of error scale" lower. The high–to–low density fade condition can be seen more readily in larger networks since many more nodes are sampled. If you detect the condition, put the output pad for the plot near the bottom of the screen. Then request another incidence density display for the "Transmit modem error" performance statistic. Set the plot resolution to the same level you used for the "Rcv CRC error" performance statistic. Move the second output pad so that its columns line up with the first. Look at the node at which the high–density fade begins in the "Rcv

crc" plot. Check for a high-density vertical line located under the same node in the "modem error" plot. In this situation, the node is producing modem errors, which corrupts data in nodes downstream. However, the further away a node is from the problem node, the fewer of its received messages must pass through the problem node, so the lower its percentage of corrupted data. The increasingly lower percentages produce the fade effect. Figure A–4 shows two plots aligned in just this manner.

If you isolate a problem node by aligning plots, investigate the node, the node upstream, and their cables and connectors very thoroughly. Use this technique with "Density across network" plots as well as incidence density plots. In addition, look for density fade conditions for "Rcv ack parity" and "Rcv header checksum" performance statistics.

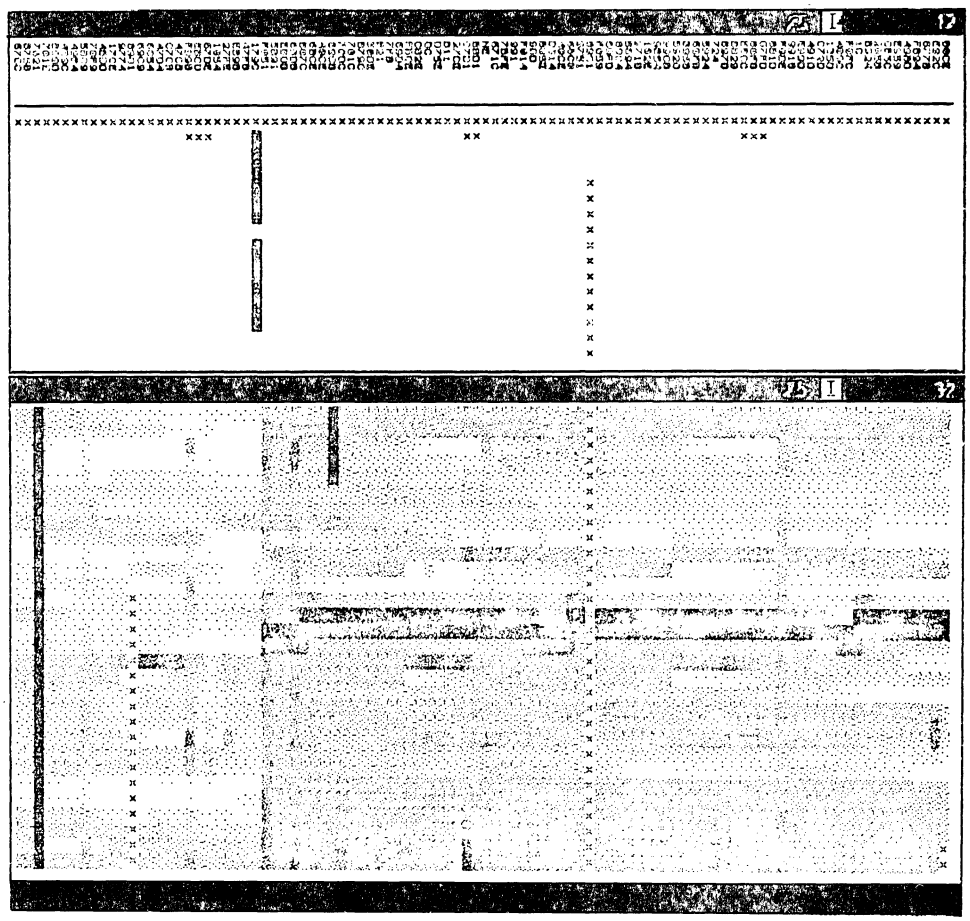


Figure A–4. Aligned Plots

More Intensive Methods of Locating Network Performance Problems

If you want more information about a node than is provided by the methods described previously, activate the SWD_10_MSGS or SWD_100_MSGS probe on a monitor in the network. These probes record performance statistic levels before and after transmission of a set of messages to each node. The SW DIAGNOSTIC (SWD) category includes many of the performance statistics that are in the RING RECEIVE category, but the counts displayed are only about packets sent by SWD messages.

To analyze SWD performance statistics, use the techniques described in the preceding sections. Be aware, however, that these probes can significantly add to network traffic. Use them to get information unobtainable by other means.

A.3 The Network Log Book

Keep a network log book, containing the information collected from **netmain_srvr**, online or as a hard copy. In it, store files produced from the **netmain** interactive tool displays and other relevant network information. Record the date and time that new nodes are installed in the network, and identify the nodes that run network services.

Include in the log book the following information about network problems:

- Date and time the problem was detected (essential in tracking network problems and helpful in interpreting the information gathered by **netmain_srvr**)
- Name of person who discovered and/or debugged the network problem
- The node that reported a ring hardware error, if any, or from the **netmain** interactive tool's Analyze Network Data menu
- The **netmain** interactive tool analysis information that relates to the problem
- Action taken to restore the network, if necessary

Write supplemental notes to **netmain_srvr** log files. Include information about scheduled service time, new node installations, and problem occurrences. Write notes in two ways:

- With the **netmain** interactive tool, use the "F4 Log a text string" command box in the Change Monitor Behavior menu.
- Outside the **netmain** interactive environment, use the **netmain_note** command, as shown below.

Type:

```
% /com/netmain_note message
```

For example, you might type:

```
% /com/netmain_note Loop 17 scheduled downtime
```

To retrieve the notes stored in a log file, use the "Printed Output" format in the **netmain** interactive tool's Analyze Network Data menu.

A.4 The netmain_srvr Reference

The **netmain_srvr** (/sys/net/netmain_srvr) program collects data used for maintaining the Apollo Token Ring network. Use the data for monitoring ring performance and isolating potential problems. Use the **netmain** interactive tool, described in Section A.5, to interact with **netmain_srvr**.

The **netmain_srvr** process running on any node is called the node's monitor. Monitors execute subprograms called "probes" and "observers", which are usually in a waiting state. At specified intervals, each probe becomes active and collects some data about nodes in the ring. The probe stores the information it collects in non-ASCII files called "log" files, then returns to its waiting state.

Each time a probe gathers data, an observer sifts through it to detect either a transmit modem error (**MODEM_ERRS**) or a disk drive error (**WIN_CRC**). If an observer detects either condition, it can provide an alarm on the node running a monitor.

The **netmain_srvr**'s probe and observer log files on active monitors are open. You cannot analyze the information in them until you close them with the **netmain** interactive tool. Log files reside in the **'node_data/system_logs/net_log** directory by default.

If netmain_srvr does not start properly, a record of the failure appears in the file **'node_data/system_logs/netmain_srvr.err.log**. If you experience problems starting **netmain_srvr** or if the monitor dies, use the data in the error log to determine the cause of the problem. The **netmain_srvr** error logs are ASCII files, and you can treat them as you would treat any ASCII file.

There are two shell commands that can help you manage **netmain_srvr** logs. The first, **netmain_chklog**, checks for and, if appropriate, deletes corrupted **netmain_srvr** log files. Use **netmain_chklog** if a node running **netmain_srvr** crashes or is reset (via the RESET switch or button). The log file that **netmain_srvr** was writing when the node crashed will be corrupted. Delete it with **netmain_chklog**. Attempts to analyze data from a corrupted log file may cause **netmain** to behave unpredictably.

Use the **netmain_note** shell command when you are outside of the **netmain** interactive tool environment and want to send a text string to a **netmain_srvr** log file. The *Command Reference* manual for any of the three environments provides information on **netmain_chklog** and **netmain_note**.

A.4.1 Invoking netmain_srvr

The **netmain_srvr** process should run as a background process from the **etc/rc** file. By default, the process names itself **netmain_srvr**, so you need not use the **-n** option with the process creation command. Run monitors only on nodes with disks, not on diskless nodes or Domain Server Processors (DSP)s. Whenever possible, run a monitor in each loop so that data can be collected even if the loop is switched out of the main ring. Ensure that monitors are configured so that they collect the data you need, without creating log files that take up too much disk space, and without affecting performance of the nodes on which the monitors run.

Online help for instructions on controlling **netmain_srvr** after it starts, and on analyzing the data collected is available by typing

% /com/help netmain

For information about adding notes to the network error log, type

% /com/help netmain_note

For information about detecting and deleting corrupt log files, type

% /com/help netmain_chklog

To start **netmain_srvr** from the DM command line, type

Command: **/etc/server netmain_srvr** [*-options*]

The process begins immediately and continues after logout.

To automatically start the server, uncomment the following four lines in the disked node's **etc/rc.user** file.

```
#if [ -f  /sys/net/netmain_srvr  ]; then
#  (echo " netmain_srvr\c" >/dev/console)
#  /sys/net/netmain_srvr &
#fi
```

The process begins when the node comes online and continues after logout. If you don't wish to use the entire set of defaults for a monitor, you can specify those you wish to use with options in the **netmain _srvr** command line or configuration file.

When started by the methods described above, **netmain_srvr** continues running until it is intentionally stopped with either of the following shell commands:

% kill netmain_srvr -q (UNIX operating system environments)

% sigp netmain_srvr -q (Aegis operating system environment)

The process stops running if the node is shut down intentionally or if the system crashes. If the process stops running, you may start it from the DM command line or by rebooting the node.

Options and Arguments

The **netmain_srvr** process has a number of options. Instead of including them all on the command line you can use an options file by specifying the **-c[mdf]** option. If you specify **-c** pathname, the server first reads the options listed in the options file specified, and then reads any other options on the **netmain_srvr** command line. If there are any conflicts between the options file and the command line, the command line settings are used. For example, if the options file specifies **-ll 1500** and the command line specifies **-ll 3000**, 3000 is the limit on the log file's length.

Default options are indicated by (D).

-a[ppend] Appends to an existing log file the name specified by the **-l** option; otherwise, creates a log file with this name. This option is only valid when a log file pathname is specified with the **-l** option. Contrast this with the **-nappend** option.

-c[mdf] [*pathname*]
Accepts options from an ASCII text file pathname. You may use this option only from the command line, not in the options file. There can only be one options file.

-l[og] [*pathname*] (D)
Creates a log file. Optionally, this specifies a pathname, which is rela-

tive to the **'node_data/system_logs/net_log** directory. If either this option or the pathname is not specified, the log filename is derived from the current date: **'node_data/system_logs/net_log/net_log.yy.mm.dd**. The log file is stored on the disk of the node running **netmain_srvr** and must remain there for **netmain_srvr** to write to it.

A.4.2 Data Collected by netmain_srvr Probes and Observers

The following list describes the data collected by the probes and observers run by **netmain_srvr** monitors. The descriptions of items sometimes refer to "output formats," or "display formats." Generate these formats with the **netmain** interactive tool.

CPU_TIME - Null/AEGIS/user CPU Time

The CPU_TIME probe records performance statistics about each node's CPU usage and writes them to **'node_data/system_logs/net_log/net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Aegis CPU time: Shows the time the operating system on each node spends on internal needs, and on servicing requests from other nodes. Expressed as a percentage of elapsed time. This statistic does not show the time the operating system spends servicing calls from user programs, running server programs, or running the Display Manager.

Null CPU time: Shows the time each node's CPU is idle, as a percentage of elapsed time. Idle CPU time is any time during which the CPU is not performing computations. Expect all nodes to show a certain amount of idle time, for time spent servicing page faults, etc. If a node shows both a high disk activity level and a high CPU time level, it may be spending too much time servicing page faults (thrashing).

User CPU time: Shows, as a percentage of elapsed time, the time the CPU on each node spends running user programs, shell programs, the Display Manager, and server processes. The statistic does not currently show the time used by individual processes or programs.

DISK_ERRS - Disk and Storage Module Errors

The DISK_ERROR probe records cumulative information about disk and storage module performance and errors on all nodes and writes them to **'node_data/system_logs/net_log/net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Controller busy: Counts the number of requests for disk I/O that could not be serviced because the controller was busy. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

CRC errors: Counts Cyclic Redundancy Check (CRC) errors on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Disk reads as percentage of disk I/O:
Calculates the ratio of reads to the node's total Winchester disk I/O. The performance statistic is shown as a percentage of the node's total Winchester I/O. If the count for this statistic is 49 percent, for example, reads account for 49 percent of all the node's Winchester I/O.

Disk writes as percentage of disk I/O:
Calculates the ratio of writes to the node's total Winchester disk I/O. The performance statistic is shown as a percentage of the node's total Winchester I/O. If the count for this statistic is 51 percent, for example, writes account for 51 percent of all the node's Winchester I/O.

Equipment check: Counts the number of device equipment checks that occur. Problems on the device controller, such as controller memory component errors, internal micro–diagnostic failures, or internal timing problems, can cause equipment checks. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Not ready: Counts the number of times that a "disk not ready" error condition occurs on a disk. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Overruns: Counts Direct Memory Access (DMA) overruns on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Seek error: Counts the number of attempts to find a track that occur on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Storage module reads as percentage of storage module I/O:
Shows the ratio of storage module reads to the node's total storage module I/O. The performance statistic is shown as a percentage of the node's total storage module I/O. If the count for this statistic is 51 percent, for example, reads account for 51 percent of all the storage

module's I/O. Only data for the first storage module (Unit 0) is displayed. Counts are expressed as a percentage of the device's I/O.

Storage module writes as percentage of storage module I/O:
Shows the ratio of storage module writes to the node's total storage module I/O. The performance statistic is shown as a percentage of the node's total storage module I/O. If the count for this statistic is 51 percent, for example, writes account for 51 percent of all the storage module's I/O. Only data for the first storage module (Unit 0) is displayed.

Timeouts: Counts the number of controller timeouts on the storage device. On storage modules, only data for Unit 0 is recorded. The display formats show the statistic as a percentage of the disk's total I/O.

Total storage module activity:
Measures the total disk storage module activity (reads plus writes) of each node (always 0 for nodes without storage modules). Use this performance statistic only with plots of network totals or rates from the **netmain** interactive tool. Shows the disk activity per node as a percentage of the total Winchester disk activity for the entire network, or absolute I/O rates per unit time. Only data for the first storage module (unit 0) is displayed.

Total Winchester disk activity:
Measures the total Winchester disk activity (reads plus writes) of each node. Use this performance statistic only with plots of network totals or rates from the **netmain** interactive tool. Shows the disk activity per node as a percentage of the total Winchester disk activity for the entire network, or absolute I/O rates per unit time.

ERR_COUNTS – Network Error Counts (Normal Traffic)

The ERROR_COUNTS probe records cumulative network performance statistics and error counts on all nodes and writes them to '**node_data/system_logs/net_log/net_log.yy.mm.dd** or a file you specify.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Acknowledge, negative (NACK):
Counts the number of transmissions in which the destination node did not acknowledge receipt of the message. NACKs can occur when nodes send messages to a node whose loop is switched out of the network, or a node that is not running the operating system. Expect large numbers of NACKs on nodes running **netmain_srvr**. The display

formats show the statistic as a percentage of the node's total network transmits.

Acknowledge parity, receive (ACK):
Counts the number of parity errors in the hardware protocol segments of received messages. The hardware always detects ACK parity errors if any occur. The display formats show the statistic as a percentage of the node's total network receives.

Acknowledge parity, transmit (ACK):
Counts the number of parity errors in the hardware protocol segments of transmitted messages. The hardware always detects ACK parity errors if any occur. The display formats show the statistic as a percentage of the node's total network transmits.

Acknowledge, Wait (WACK):
Counts the number of times that the destination node was too busy to accept a message in the time allotted. This performance statistic does not necessarily reflect an error condition — on a DN4xx/DN6xx nodes, for example, the node may have been performing disk I/O using the shared ring/disk hardware. The display formats show the statistic as a percentage of the node's total network transmits.

Biphase error, receive:
Counts the number of biphase errors that occur. Biphase errors occur when a node's modem hardware cannot lock on the transmission frequency from the node upstream. Biphase errors can result from modem hardware failures, broken cables or connectors, or signal degradation caused by excessive cable lengths between active nodes. The display formats show the statistic as a percentage of the node's total network receives. The ERR_COUNTS probe collects data for this statistic only from nodes running SR8 or later releases.

Biphase error, transmit:
Counts the number of biphase errors that occur. Biphase errors occur when a node's modem hardware cannot lock on the transmission frequency from the node upstream. Biphase errors can result from modem hardware failures, broken cables or connectors, or signal degradation caused by excessive cable lengths between active nodes. The display formats show the statistic as a percentage of the node's total network transmits. The ERR_COUNTS probe collects data for this performance statistic, only from nodes running SR8 or later releases.

Bus error, receive: Counts errors during Direct Memory Access (DMA) from the ring controller. The display formats show the statistic as a percentage of the node's total network receives.

Bus error, transmit:

Counts the number of times that the ring controller experienced a bus error during a Direct Memory Access (DMA) transfer. The display formats show the statistic as a percentage of the node's total network transmits.

CRC, receive:

Counts messages that contain Cyclical Redundancy Check (CRC) errors, detected by the ring hardware. CRC errors can result from errors in any part of the received message, except the hardware protocol segments. You cannot disable CRC checking. Compare CRC error statistics to those for RCV header checksum and ACK parity errors. The display formats show the statistic as a percentage of the node's total network transmits.

End-of-range, receive:

Counts the number of times that one or both of the message fields in the packet received was larger than the Direct Memory Access (DMA) channel allowed for it. The display formats show the statistic as a percentage of the node's total network receives.

ESB errors, receive:

Counts the number of elastic store buffer errors received. These errors arise when the node cannot follow a large or sudden change in the network communication frequency. The display formats show the statistic as a percentage of the node's total network receives. The ERR_COUNTS probe collects data for this statistic only from nodes running SR8 or later releases.

ESB errors, transmit:

Counts the number of Elastic Store Buffer (ESB) errors that occur. ESB errors occur when the node is unable to follow a large or sudden change in the network's communication frequency. The display formats show the statistic as a percentage of the node's total network transmits. The ERR_COUNTS probe collects data for this statistic, only from nodes running SR8 or later releases.

Header checksum, receive:

Counts messages that contain checksum errors in the message header. Since the operating system program that verifies header checksums is usually disabled, the count for this statistic should be 0. The display formats show the statistic as a percentage of the node's total network receives.

Modem error, receive:

Counts the number of times the receiver could not synchronize properly with the network. The conditions that cause this error are biphase or Elastic Store Buffer (ESB) errors. See the "receive biphase error" or "receive ESB error" statistics. The display formats show the statistic as a percentage of the node's total network receives.

Modem error, transmit:

Counts the number of times the transmitter could not synchronize properly with the network, resulting in an Xmit ESB or biphase error condition. See "transmit biphase errors" and "transmit ESB errors." If the error condition lasts more than one minute, the node broadcasts a "hardware failure report" (displayed in the nodestat shell command output). A broken cable can cause modem errors. The display formats show the statistic as a percentage of the node's total network transmits.

No return, transmit:

Counts the number of transmitted messages that failed to return to the node. After a destination node receives and copies a message, the message continues to travel around the ring until it returns to its transmitter, which removes the message. If a message does not return, it could not complete the loop around the ring, indicating a break in the ring. The display formats show the statistic as a percentage of the node's total network transmits.

Overrun, receive: Counts the number of Direct Memory Access (DMA) overruns that occur. The display formats show the statistic as a percentage of the node's total network receives.

Overrun, transmit: Counts the number of times that the ring controller experienced a Direct Memory Access (DMA) overrun condition during a ring transmit. The display formats show the statistic as a percentage of the node's total network transmits.

Packet error, receive:

Counts the number of times either the receiver or the transmitter had problems with messages. If the fault was in the receiver, other counts are incremented also. The display formats show the statistic as a percentage of the node's total network receives.

Packet error, transmit:

Counts the number of times an error occurs during transmission of a message. The system increments counts for this statistic when other error conditions occur. The display formats show the statistic as a percentage of the node's total network transmits.

Timeout, receive: Counts messages received that did not complete in the expected time. The display formats show the statistic as a percentage of the node's total network receives.

Timeout, transmit: Counts transmitted messages that do not complete their transmission in the expected time. This error often occurs when network traffic is slow, due to repeated attempts to retransmit or regenerate the ring token. The display formats show the statistic as a percentage of the node's total network transmits.

Transmit call:
: Counts the number of requests to transfer data out of the node, on to the ring. The transmit call counter is increased even if the actual transmit fails. This performance statistic does not reflect any error conditions. If the number of requests is less than 100 percent of the ring transfers attempted, the node had to retry some of the transfers. The display formats show the statistic as a percentage of the node's total network transmits.

Transmit error, receive:
: Counts the number of times that either the transmitter or another receiver had an error in the packet. For this error to occur, some other error flag must be set. The display formats show the statistic as a percentage of the node's total network receives.

Receives as percentage of network I/O:
: Calculates the ratio of incoming messages (receives) to the node's total network I/O. The performance statistic is shown as a percentage of the node's total network traffic. If the count for this statistic is 43 percent, for example, incoming messages account for 43 percent of all the node's network I/O activities.

Sends as percentage of network I/O:
: Calculates the ratio of transmitted messages (sends) to the node's total network I/O. The performance statistic is shown as a percentage of the node's total network traffic. If the count for this statistic is 43 percent, for example, outgoing messages account for 43 percent of all the node's network I/O activities.

Total network activity:
: Measures the total network activity (sends plus receives) of each node. You should use this performance statistic only with plots of network totals or rates. Total network activity per node is shown as a percentage of the total network activity for the entire network.

EST_TOPOLOGY - Topology Information

The EST_TOPOLOGY probe writes information collected by the TOPOLOGY probe (see below) to the **netmain_srvr** log file.

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: 1

HW_FAIL - Hardware Failure Messages

The HW_FAIL probe records every change in the hardware failure message reported by the **nodestat** command, on the node that is running the monitor. The "ring hardware fail-

ure" message identifies the node that last reported a ring hardware failure. Use the message to locate the problem node or cable in the network. The node that reports the failure is often contiguous to the failure or is experiencing the failure.

Default Probe Interval Time: 0:01:00

Default Probe Skip Distance: Not Applicable

Hardware failure messages are generated by Domain/OS. They appear when there is no network traffic, and Domain/OS is completely unable to send network messages.

MEMORY – Records Counts of Memory Errors on Nodes in the Network

The MEMORY probe lists nodes on which correctable memory errors have occurred.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

NET_SERVICE – Network Service Queue Statistics

The NET_SERVICE probe measures the length of the network service queue backlog on each node. The length is the number of network service requests that remain in the queue immediately after a request is serviced.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

File server, any backlog:

> Shows the number of times that each node's file server noted one or more file service requests in the file server queue. Any requests left in the queue create a queue backlog. The file server checks for a backlog every time it services a request. The file server handles only requests from other nodes for operations such as opening, closing, and creating files (not reading or writing). Output formats show the number of times a backlog was present as a percentage of the number of file services requested by other nodes.

File server, average backlog:

Shows the average number of file service requests in each node's file server queue. Every time it services a request, the file server checks for remaining requests (the queue backlog) and counts these remaining requests. When there are no requests, the performance statistic adds a 0 to the average. The file server handles only requests from other nodes for operations such as opening, closing, and creating files (not reading or writing). Backlog incidence is shown as a percentage of the queue's capacity.

File server, backlog severity:

Shows the average length of each node's file service backlog. Every time it services a request, the file server checks for remaining requests (the queue backlog). If a backlog exists, the number of requests the queue contains are used for averaging in this performance statistic. The file server handles only requests from other nodes, for operations such as opening, closing, and creating files (not reading or writing). Incidence is shown as a percentage of the queue's backlog capacity.

File service queue overflow:

Shows the number of rejected requests for file services for other nodes. A node rejects file service requests when it already has too many other requests pending. Rejections slow down the node that made the request and can cause errors. When a node has service queue overflows, it can indicate that too many other nodes require data stored on this node. Output formats shows this statistic as a percentage of the file service requests made to each node.

Total file service: Shows how many file services each node performs for other nodes (not file services the node performs for its own benefit). You should use this performance statistic only with plots of network totals or rates. File services are activities such as opening and creating files, and directory lookups. (The paging server handles file reads and writes.)

Paging server, any backlog:

Shows the number of times that each node's paging server noted one or more page service requests in the page server queue. Any requests left in the queue create a queue backlog. The paging server checks for a backlog every time it services a request. The paging server handles only requests from other nodes, for reads, writes, paging, and several internal operating system services. Incidence plots show the number of times a backlog was present as a percentage of the number of page services requested by other nodes.

Paging server, average backlog:

Shows the average number of page service requests in each node's page server queue. Every time it services a request, the paging server checks for remaining requests (the queue backlog) and counts these remaining requests. When there are no requests, the performance statistic adds a 0 to the average. The paging server handles only requests from other nodes for reads. Backlog incidence is shown as a percentage of the queue's capacity.

Paging server, backlog severity:

Shows the average length of each node's paging queue backlog. Every time it services a request, the paging server checks for remaining requests (the queue backlog). If a backlog exists, the number of requests the queue contains are used for averaging in this performance statistic. The paging server handles only requests from other nodes, for reads, writes, paging, and several internal operating system services. Incidence is shown as a percentage of the queue's backlog capacity.

Total paging service:

Shows how many paging services each node performs for other nodes (not file services the node performs for its own benefit). You can use this performance statistic only with plots of network totals or rates. Paging services are activities such as reads, writes, normal paging, and some internal operating system services. (The file server handles file opening, file creations, and directory lookups).

Reads requested: Shows the number of pages that each node has read from other nodes in the network. Nodes read pages during file I/O or any other paging activity. Output formats that use percentages show the statistic as a percentage of all the node's reads and writes to other nodes.

Reads serviced: Shows the number of pages on each node that have been read by other nodes in the network. Nodes read pages during file I/O or any other paging activity. Output formats that use percentages show the statistic as a percentage of all services the node performs for other nodes.

Writes requested: Shows the number of pages each node has written to other nodes. Nodes write pages during file I/O, operating system execution, and many other activities. Output formats that use percentages show the statistic as a percentage of all the node's reads and writes to other nodes.

Writes serviced: Shows the number of pages written to each node, by other nodes. Nodes write pages during file I/O, operating system execution, and many other activities. Output formats that use percentages show the statistic as a percentage of all services the node performs for other nodes.

PAGING – Diskless/Partner Information

The PAGING probe records information about diskless nodes and their paging partners.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

Diskless nodes/paging partners:
: Produces a display of diskless nodes and their paging partners. This display cannot be used with data taken from a running monitor. Use it to analyze data from log files.

Paging partners, ordered by diskless node:
: Shows the node(s) used as paging partner(s) by each diskless node.

Paging partners, ordered by mother node:
: Shows the diskless node(s) supported by each node that that volunteered as a paging partner.

SWD_10_MSGS – Software Diagnostic Messages (10)

The SWD_10_MSGS records counts for various performance statistics on a node, both before and after a 10–message broadcast. The data collected reflect the effect of receipt of messages on the node's performance statistics.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD ack parity:
: Counts the number of parity failures in the hardware protocol segments of software diagnostic messages. The hardware always detects ACK parity errors if any occur. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD bus error:
: Counts errors during Direct Memory Access (DMA) from the ring controller, during receipt of software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received by the node.

SWD CRC: Counts test messages that contain Cyclical Redundancy Check (CRC), errors, detected by the ring hardware. CRC errors can result from errors in any part of the received message except the hardware protocol segments. Note that this performance statistic shows only those errors that occur in software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collects data for this performance statistic. The display formats show the statistic as a percentage of the diagnostic messages received.

SWD end–of–range:
Counts the number of times that one or both of the message fields in the test message received was larger than the Direct Memory Access (DMA) channel allowed for it. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD header checksum:
Counts SWD (Software Diagnostic) messages that contain checksum errors in the message header. Since the operating system program that verifies header checksums is usually disabled, the count for this statistic should be 0. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD messages not received:
Shows the percentage of Software Diagnostic messages sent to a node but not received, as a percentage of the total number of SWD messages sent to the node. The SWD_10_MSG and SWD_100_MSG probes collect the data for this performance statistic.

SWD modem err:
Counts the number of times the receiver could not synchronize properly with the network, during receipt of SWD messages. The conditions that cause this error class are biphase or Elastic Store Buffer (ESB) errors. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the SWD test messages the node receives.

SWD overrun: Counts the number of Direct Memory Access (DMA) overruns that occur during software diagnostic messages. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD packet error: Counts the number of times either the receiver or the transmitter had problems in SWD (Software Diagnostic) messages. If the fault was in the receiver, other counts are incremented also. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages the node receives.

SWD receive timeout:
Counts Software Diagnostic (SWD) messages received that did not complete in the expected time. The SWD_10_MSG and SWD_100_MSG probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received.

SWD transmit errors:
Counts the number of times that either the transmitter or another receiver had an error in a Software Diagnostic test message. For this error to occur, some other error flag must be set. The SWD_10_MSG and SWD_100_MSGS probes collect data for this performance statistic. The display formats show the statistic as a percentage of the test messages received by the node.

SWD_100_MSGS – Software Diagnostic Messages (100)

The SWD_100_MSGS records the effect of 100–message broadcasts, in the same manner as SWD_10_MSGS above. This probe collects the same information as SWD_10_MSGS.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

TIME_SKEW – Difference Between Node Clocks

The TIME_SKEW probe detects differences between the node clocks.

Default Probe Interval Time: 3:00:00

Default Probe Skip Distance: 1

Compute offset times:
Automatically computes offset times for each log file. The computed offset times are derived from the contents of the log files, using results from the TIME_SKEW probe. A variety of conditions can prevent the offset computation from working. For instance the TIME_SKEW probe may never have executed or may not have operated on each node.

TOPOLOGY - Total Node List Estimate

The TOPOLOGY probe provides total node count (estimate).

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: Not Applicable

Topology list from node total:

A monitor keeps an estimate of the complete ring topology. The estimate represents a running total of lcnode results, not just the most recent result. This topology list will probably be longer than the list produced by the lcnode command, especially if the monitor has been running for a long time. In particular, it may list nodes that are not currently running. See also "EST_TOPOLOGY" above.

MODEM_ERRS - Transmit Modem Errors

This observer reports on nodes that have more than five times the average number of "Transmit Modem Errors."

Default Probe Interval Time: 0:30:00

Default Recheck Interval: 12:00:00

WIN_CRC - Disk Drive Errors

This observer reports on nodes that have more than 0.01 percent of Winchester disk drive CRC errors.

Default Probe Interval Time: 0:30:00

Default Recheck Interval: 12:00:00

A.4.3 Using netmain_srvr to Build Topology Lists

The **netmain_srvr** and the **netmain** interactive tool provide several kinds of topology lists for the Apollo Token Ring network, including a Master Topology List. The Master Topology List includes:

- The direction of data flow in the network: the monitor from which information is collected and all nodes downstream of that node
- All nodes whether or not they are currently connected to the network

- Each node's hexadecimal ID
- Diskless nodes

Additional information that should be added to this list includes

- Loop number
- Node type
- Paging partners of diskless nodes
- Network servers running on the node
- Node administrator's name (if they occur at your site)
- Node location

When complete, this list is the network's Master Topology List. The Master Topology List provides the foundation for allocating network resources and for monitoring and "tuning" network performance.

The netmain_srvr Topology Lists

When you run **netmain_srvr**, the server process creates a Total Node List. The list is **netmain_srvr**'s best estimate of what the network topology would look like if every node in the network were switched on. To generate the list, **netmain_srvr**'s TOPOLOGY probe performs the equivalent of the **lcnode** command at scheduled intervals.

Under normal operating conditions, if the TOPOLOGY probe runs for several days to a week, it produces a complete network topology list. Since it is repeated at scheduled intervals, the TOPOLOGY probe may find a node at some probe that it did not not find in previous probes. The node may have been newly installed, its loop may have been switched out during a previous probe, or it may have been powered down or not communicating on the network. The TOPOLOGY probe adds newly acquired node information to **netmain_srvr**'s Total Node List.

The EST_TOPOLOGY probe writes **netmain_srvr**'s Total Node List to the **'node_data/system_logs/net_log** directory. The directory contains logs from all **netmain_srvr** probes and observers. The **netmain_srvr** uses non-ASCII files for its work. You can look at the files in **'node_data/system_logs/net_log** by using the netmain interactive tool. In fact, this tool provides the only means you have of looking at any information in **netmain_srvr**'s **net_log** files. (You can look directly at error logs.)

To generate the Master Topology List, run **netmain_srvr** with its default options. If you use the **netmain** interactive tool on the monitor while you are collecting a topology list, be sure the TOPOLOGY and EST_TOPOLOGY probes remain active. After you have run **netmain_srvr** for several days to a week, enter the **netmain** interactive environment from the shell. Choose the Find Monitors and Nodes menu. Use "F4 Topo list from node total" to cause **netmain_srvr**'s Total Node List to be written to the **netmain** environment. When the topology list status message tells you the list is written, you can use "F6 Display topology list" to display the topology list. Verify that the total of nodes on the list equals the total of nodes in your network. If it does not, let the monitor run longer. Store the topology list on line and post printed output in the network control room. Section A.5 describes how to save displays generated with the **netmain** tool. Add the additional information, listed above, to the topology list after storing the list as an ASCII text file.

The **netmain** tool can also generate a topology list by executing the **lcnode** command once. From the Find Monitors and Nodes menu, use the "F3 Topo list" from **lcnode** command to generate the list. Use the F6 command to display the list. This topology list has the same uses, advantages, and drawbacks as those described for the **lcnode** command.

In large networks, subsets of the Master Topology List are often useful. A subset of the master list might contain, for example, only DN3000 nodes. Different types of nodes perform differently. Special purpose topology lists can readily show the kinds of nodes that are experiencing problems or unexpected patterns of use. Use special-purpose topology lists and performance statistics to improve network performance.

If your network extends through several buildings, use subsets of the Master Topology List to identify all nodes in a single building. Use the list to identify conditions applicable only to that building, for example, to isolate problems caused by interference in power supply.

Finally, you should have a topology list of all nodes running monitors in the network. The F2 command "Find all monitors" will generate this list.

A.4.4 Using netmain_srvr to Gather Performance Statistics

The topography and topology lists provide base-level information about your network. Set up a system of regularly scheduled reports from **netmain_srvr** monitors to build a picture of the network's performance over time.

The **netmain_srvr** is the main source of network performance statistics. Shell commands can provide limited information on the performance of nodes and the network. Use **netmain_srvr** as part of an ongoing network maintenance effort because it is the only means of collecting information about

- Events as they occur over time
- The time at which an event count begins

- An event's pattern of occurrence
- Events that occur regularly, but at a low frequency (for example, 1 in 10,000 times)

In addition to generating topology lists, **netmain_srvr** collects more than 75 different statistics (defined earlier in this section) on node and network performance and error conditions. You control **netmain_srvr** probes and observers with **netmain_srvr** configuration files or the **netmain** interactive tool. Both allow you to specify

- What data is collected
- How often the data is collected

The **netmain** interactive tool also allows you to

- Change specifications on active monitors
- Choose a variety of output formats for the data

(See Section A.5 for more information on the **netmain** tool.)

Relationship of netmain_srvr Probes to Network Topology

Information on the ring originates at a transmitting node or source. The source node sends data by placing a message, or packet, on the ring. The packet contains the destination address of the node that the packet is for.

When it receives a packet, a node examines the packet's destination address. If the destination address matches the node's ID, the node

- Copies the packet into its memory
- Modifies the packet to acknowledge successful (or unsuccessful) receipt
- Sends the modified packet to the next node

When a node receives a packet with a destination address that does not match its own ID, it sends the packet to the next downstream node. When a packet returns to its source node, the source node removes the packet from the ring and examines the packet to determine if it was successfully received.

The **netmain_srvr** probes report on the state of

- The hardware that sends and receives packets, specifically the ring controller and modem
- The packet transmission protocols, the condition of the packets sent and received, the number of packets delayed or lost, and the total number of packets sent and received

A node may be unable to send or receive messages if its ring hardware is not functioning properly. If some part of the route between the source and destination nodes is completely broken, the message will never arrive at the destination node.

A packet can leave a node only if it has the correct format. However, a message can arrive at the destination node in a corrupted state for several reasons. Frequently the cause is a weak signal or intermittent signal interference on the coaxial cable. All nodes cause momentary instability on the ring when they are put into the network. That momentary instability can cause a packet to become corrupted.

Both serious and minor conditions are reflected in network data. A ring hardware failure is serious and requires immediate attention. Other conditions can slow down overall network performance but do not cause complete network failure.

All nodes monitor the condition of packets whether or not the packet is intended for the node. Error conditions indicating hard breaks in the network are detected by using the broken-link detection mechanism. Nodes that are downstream of a break report the failure; source nodes upstream of a break are unable to send messages past the broken link.

In addition to the data described above, other **netmain_srvr** probes report on both normal and potentially problematic performance of storage volumes.

Probes Reporting Error Conditions

The probes named in Table A-1 detect potentially serious node and network error conditions. Any of these errors can happen at random, in short bursts. In these cases, the cause is usually a momentarily weak signal. However, if the counts of these probes show a steady increase, there is reason to suspect a problem with a disk, a node, or network cable.

Table A–1. netmain_srvr Probes Reporting Serious Error Conditions

Probe	Level of Error
DISK_ERRS	
CRC errors	Count should not be higher than 0.01%
Seek errors	Count should not be higher than 0.01%
Equipment check	Count should not be higher than 0.01%
ERR_COUNTS	
Receive Header Checksum	Count should be 0
Biphase Error	Can indicate hard break in network
ESB Error	Can indicate hard break in network
Modem Error	Can indicate hard break in network
Transmit No Return	Can indicate hard break in network
HW_FAIL	Indicates hard break in network

Probes Reporting on Network Performance

The **netmain_srvr** probes track the services nodes perform for each other. These are

- Paging service—transferring 1024-byte pages of objects from one node to another
- File service—requests from remote nodes to create, open, or close a file on the local node
- Reading pages on and writing pages to other nodes
- Serving as a paging partner for diskless nodes

Node CPU and disk activity is closely related to the services nodes perform for each other; **netmain_srvr** also collects information on this activity. Table A–2 shows the **netmain_srvr** node and network performance statistics that can be useful for allocating network resources.

Table A-2. netmain_srvr Probes Reporting on Network Performance

Probe	Event to Monitor
CPU_TIME	Aegis CPU User CPU
DISK_ERRS	Total Storage Module Activity Total Winchester Disk Activity
NET_SERVICE	File Server Backlog Paging Server Backlog Reads/Writes Requested File Server Queue All Paging Server Queue Reads/Writes Serviced Total File Service Total Paging Service
PAGING	Monitor all events generated by this probe

Controlling netmain_srvr's Data Collection Characteristics

The **netmain_srvr** can collect a large amount of data, but you can limit the number of active probes on each monitor if there is a shortage of disk space in the network or if you simply find it easier to do so. For example, you might configure a monitor to collect only topology data, another to collect network performance statistics, and another to collect error statistics.

Decrease a probe's sampling interval time and skip distance; that is, cause the probes to collect information more often, especially when you suspect a problem with a node, a loop, or the network. Decrease the default schedules if you need greater detail about some network performance events or if your service representative requests the information for network maintenance purposes. The default probe sampling intervals are sufficient for most network data collection purposes.

Use the **netmain** interactive tool's Alter Probe Timing submenu to change sampling intervals and skip distance on an active monitor. In **netmain_srvr** configuration files, **-observer** and **-sample** options control the sampling interval time. The **-skip** option controls the skip distance.

You can explicitly set a limit on the length of the log files written by **netmain_srvr**. Use the **-ll** (log length) option in **netmain_srvr** configuration files. The **netmain** interactive tool's Alter Logging Controls submenu allows you to limit log file size. For regular network data collection, it is good practice to

- Limit the length of log files; the default length is usually adequate, but verify this by regularly inspecting log size when you first start monitors or change monitor parameters.

- Automatically start new log files after you close old log files; this is the default and should not be overridden by the configuration file option **-nl**.

- Use the Alarm Server **-disk** [*n*] option on user nodes running monitors. A user logged on to that node will receive notice of a potential disk overflow condition. The Alarm Server startup is explained in Chapter 3 of this book.

Limiting the number of probes on each monitor, closing the log files, and opening new ones frequently (for example once a week) is usually sufficient to ensure that **netmain_srvr** does not overwrite data because disk space is used up or log length is reached. After starting **netmain_srvr** processes on nodes, check the log files written by the servers on a regular basis, typically once a week. Copy old **netmain_srvr** log files to floppy disk or tape for later analysis and permanent storage.

A.5 The netmain Interactive Tool Reference

This section describes **netmain**, the interactive tool used to manage the network and to diagnose problems with individual nodes or disks. It provides complete reference information on the tool and describes how to invoke and use it.

The **netmain** interactive tool can manage **netmain_srvr** monitors. The **netmain_srvr** monitors collect information used to allocate network resources and diagnose network problems. Most of this sections contains reference information on all **netmain** menus. The contexts in which to use **netmain** are described in Section A.1 of this Appendix.

Refer to this description of **netmain** when you set up a network management program and when you work with the tool. If you have never used **netmain**, practice with the tutorial at the beginning of this Appendix (Procedure A–2). The tutorial does not give detailed information about the menu commands. Its purpose is to allow you to become familiar with controlling the the **netmain** menus. Read the descriptive information in this Section on **netmain** before or after you practice with the tutorial.

You can configure **netmain_srvr** from the command line (see Section A.4), or you can start a server process and use the **netmain** interactive tool to configure it. The menu and submenu descriptions below reference the server options controlled from each menu.

When **netmain_srvr** is running, the **netmain** lets you control **netmain_srvr** monitors, change parameters, and analyze the information in the log files. In addition, **netmain** provides output formats and graphic displays of the information collected by the **netmain_srvr** process. The tool provides online help about menu usage and about every menu item.

A.5.1 Invoking netmain

Use the shell command **ps** (UNIX operating system command) **or pst** (Aegis command) to verify that a monitor is running before you start using **netmain**. The tool can find monitors that have been running for approximately 30 seconds. If the tool cannot locate it, the monitor may be experiencing a problem.

Invoke **netmain** from the shell command line as follows:

```
% netmain
```

You can run the **netmain** interactive tool in an icon window by selecting the F6 box in the Top Level menu and using the space bar to create the icon. Open the window from its icon mode by placing the cursor in the icon and pressing the space bar once.

When you use **netmain**, select a **netmain_srvr** monitor to work with before using any other feature of the program.

A.5.2 The netmain Top-Level Menu

When you invoke **netmain**, the program displays its Top-Level menu, shown in Figure A-5. The commands on the menu are described in Table A-3. Use the Top-Level menu to

- Invoke the other menus
- Run **netmain** in an icon window

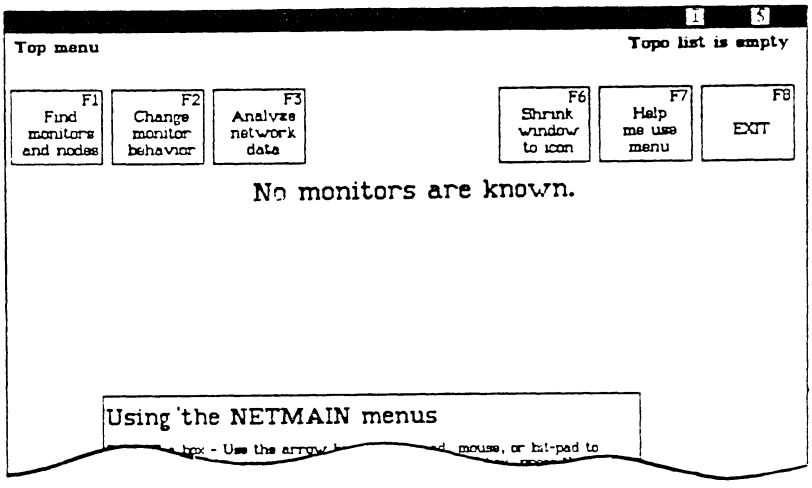


Figure A-5. Top-Level Netmain Menu

Table A–3. Top–Level Commands

Command Box	Description
F1 – Find monitors and nodes	Invokes a submenu that selects a single monitor from which to collect information or generate reports about monitors or nodes. Use this submenu to select monitors or to create and manipulate network topology lists.
F2 – Change monitor behavior	Invokes a submenu that redefines the **netmain_srvr** parameters. Use this submenu to change the behavior of an active monitor or to close an active log file for examination.
F3 – Analyze network data	Invokes a submenu that formats data and displays it on the screen. You can copy formatted data to a file for printing at a later time. Use this submenu to format data from an executing monitor or from a log file that has been closed.
F6 – Shrink window to icon	Runs the program in an icon window.
F7 – Help me use menu	Provides online help.
F8 – Exit	Quits the program.

A.5.3 The netmain Find Monitors and Nodes Menu

Figure A–6 shows the Find Monitors and Nodes menu. Use the Find Monitors and Nodes menu to

- Find monitors
- Select monitors for use
- View network topology lists

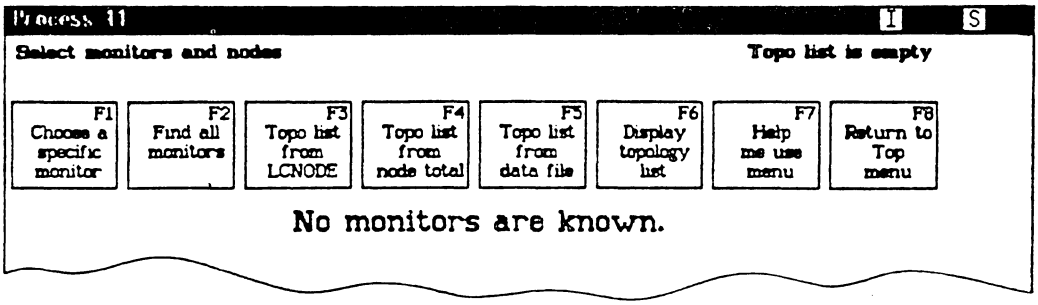


Figure A–6. Find Monitors and Nodes Menu

The F1 and F2 command boxes in this menu find specific monitors. The F3 through F6 command boxes write to and display the topology list. Table A–4 explains each command box in the Find Monitors and Nodes menu.

Table A–4. Find Monitors and Nodes Commands

Task	Command Box Used	Explanation
Select a monitor for examination or configuration	F1—Choose a specific monitor	Selects a monitor for work or analysis. Use this option when **netmain** reports "No Monitors are Known," to examine the log file currently open for this monitor; or the monitor itself; or to reconfigure the monitor. Use the input to specify the monitor by following the guidelines in the help box. **Netmain** finds the monitor and draws a box containing that node's name. If it can't find the monitor, **netmain** reports "Node appears not to have a monitor." This can happen if • The monitor is busy and cannot respond • The monitor has stopped running • The network is broken so that messages are not transmitted

Table A–4. Find Monitors and Nodes Commands (Cont.)

Task	Command Box Used	Explanation
		If the monitor is busy, try the command a second time if you receive this message. You cannot use other **netmain** features until you choose a monitor.
List all monitors	F2—Find all monitors	Finds all monitors, using the current topology list (see F6). If the topology list is empty, **netmain** executes **lcnode** and writes the results to the Topo list. Status messages appear as the program searches for monitors. **Netmain** draws a box containing the name of each monitor it finds. Select a monitor by pointing to the box. **Netmain** encloses it with stripes.
Fill topology list with current **lcnode** information	F3— Topo list from **lcnode**	Executes **lcnode** and writes the information to the Topo list. To display the list, use the F6 command box.
Fill topology list with an estimated of all the nodes physically connected to the network — even those not communicating on the network	F4—Topo list from node total	Takes the Total Node list stored in the monitor selected and writes it to the Topo list. Since the Total Node list is a cumulative list of nodes that the monitor found after several **lcnode**s, it should be the most complete online list of all the nodes in the network. When the monitor has run for a week, the Total Node list should include all nodes that have been on the network since the monitor started. The list may be longer than the current **lcnode** report. If you are using the monitor on the local node, **netmain** gets a Total Node list without generating network traffic. Use this command on a

Table A–4. Find Monitors and Nodes Commands (Cont.)

Task	Command Box Used	Explanation
		local monitor when you need to generate a topology list during network problems. To display the topology list, use the F6 command.
Fill topology list from a file	F5—Topo list from data file of nodes	Prompts for a filename and list writes its contents to a topology list. Use a log file or create a text file. If you create a text file, list node names or hexadecimal IDs, separated by spaces, or listed on different lines. Comment lines must start with the characters # or {, and the text must run to the end of the line. Use this command to build a Master Topo list or special purpose lists, for example, a list of nodes that run **netmain_srvr** or any of the network servers. The help box shows how **netmain** resolves relative pathnames. To display the contents of the topology list, use the F6 command box. If you don't enter a filename in response to the prompt, **netmain** displays an error message.
View current network or topology kept in stored topology list.	F6 — Display Topo list	Displays the current contents of the topology list in a read-only pad. You can use other commands (F3, F4, F5) to fill the topology list with particular information. If you have not used one of these other commands yet, **netmain** performs an **lcnode**, fills the topology list with the results, and displays the contents of the list. To save the list, use the DM **pn** command.

A.5.4 The netmain Change Monitor Behavior Menu

The Change Monitor Behavior menu allows you to manipulate monitor behavior. The **netmain_srvr** option **-topo [init]** corresponds to the F6 command in this menu. In order to use this menu, you must select a monitor by using the Find Monitors and Nodes menu. Then you can use this menu to

- Close a current log file so you can analyze its contents
- Schedule probes to run more frequently
- Reschedule observers
- Configure data collection parameters for each monitor

Figure A-7 shows the Change Monitor Behavior menu, and Table A-5 lists each command box in the menu.

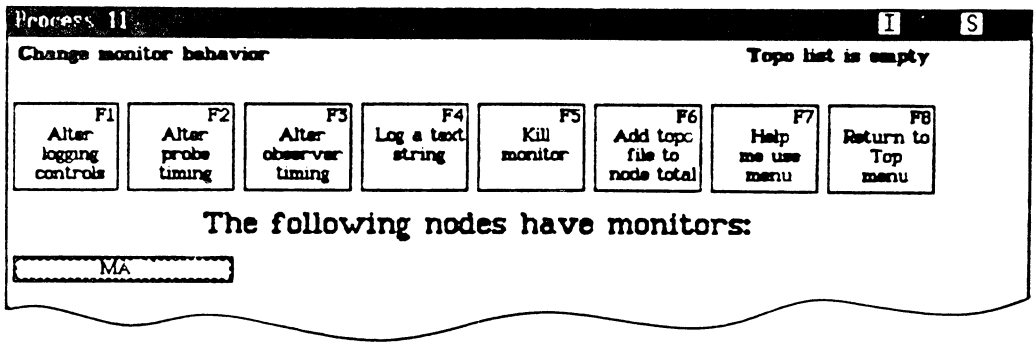


Figure A-7. Change Monitor Behavior Menu

Table A-5. Change Monitor Behavior Commands

Task	Command Box Used	Explanation
Set parameters for log file	F1—Alter logging controls	Invokes as submenu that starts new log files, which allow you to change logging parameters.
Set parameters for probes, list probes	F2—Alter probe timing	Invokes a submenu that reschedules probe active and waiting periods and probe skip distances.

Table A-5. Change Monitor Behavior Commands (Cont.)

Task	Command Box Used	Explanation
Set parameters for observer and list observer	F3—Alter observer timing	Invokes a submenu that reschedules the intervals at which observers wake up to check for conditions.
Send a text message to a monitor	F4—Log a text string	Prompts for a line of text that can contain up to 80 characters and sends the text to the selected monitor. To select a monitor, point to the box that contains its name. **Netmain** encloses the box with stripes. If the monitor name box is not displayed, return to Find Monitors and Nodes menu and choose "F2—Find all monitors".
Stop a monitor	F5—Kill monitor	To stop a monitor from **netmain**, point to the box that contains its name. **Netmain** encloses the box with stripes. If the monitor name box is not displayed, return to a Find Monitors and Nodes menu, choose "F2—Find all monitors". Note that once you stop the monitor, you must restart it.
Update a monitor's node list when the TOPOLOGY probe cannot	F6—Add topo file to node total	Updates the chosen monitor's Total Node from a topology list in a text or log file. Use this feature only if the network failed and the TOPOLOGY probe on a monitor you just started cannot get topology information to update the list. This feature should not be used at other times because, if the topology list used does not include all nodes, the monitor will not collect data from any nodes not in the topology list. The help box shows how **netmain** resolves relative pathnames.

Using the Change Monitor Behavior Menu

When you are using the Change Monitor Behavior menu, do not use CTRL/Q (Quit a process) between the time you select the "Perform changes" commands and the time the message "Node_name acknowledged" appears. Doing so may cause unpredictable monitor behavior.

A.5.5 The netmain Alter Logging Controls Submenu

The Alter Logging Controls submenu displays the parameters in effect for the log file currently in use. There are two versions of this menu, the one displayed depends on the monitors logging behavior. The parameters are the same as the following **netmain_srvr** options.

Option	Meaning
-l|-nl	Write or do not write to a log file
-l [pathname]	The log file name
-ll n	Limit the size of the log file

Use the Alter Logging Controls submenu to

- Change the parameters in effect for a log file, or
- Cancel changes and return to the parameters most recently in effect

The Alter Logging Controls submenu is shown in Figure A-8. Table A-6 lists each command box in the submenu.

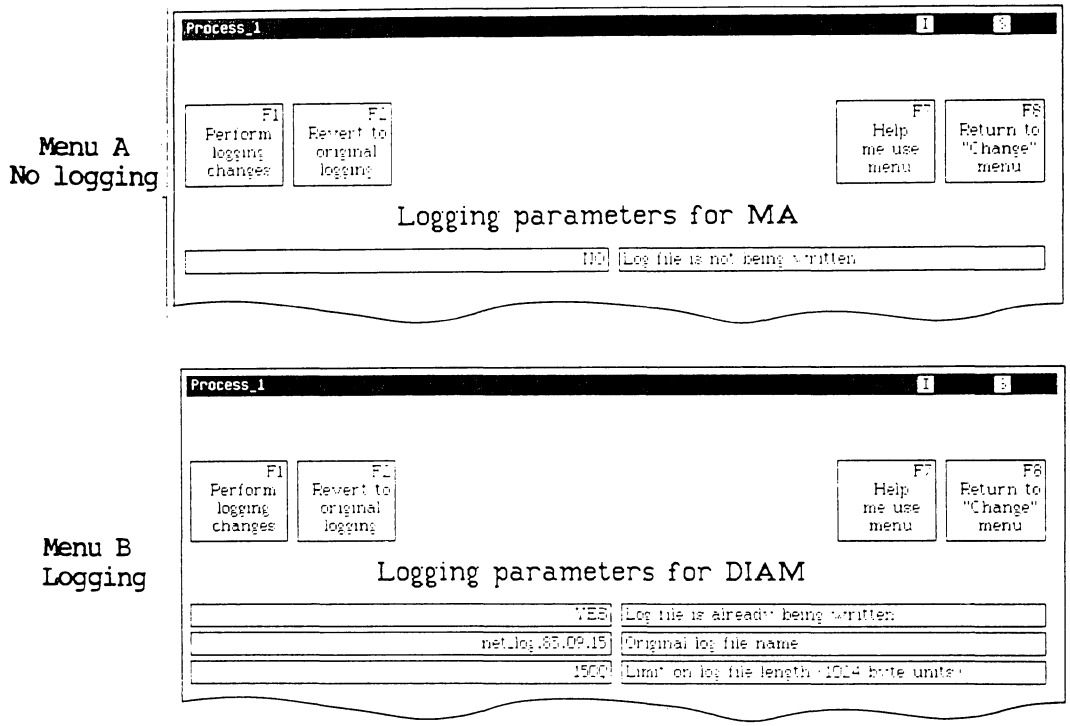


Figure A–8. Alter Logging Controls Submenu

Table A–6. The netmain Alter Logging Controls Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform logging changes	Causes the changes you make to take effect.
Cancel changes to	F2—Revert to original	Abort the changes. When you use the Alter Logging Controls sub-menu, you make changes on a "scratch pad." Select F1 to make the changes. Select F2 to abort the changes. Changes take effect when the status message, "Node_name acknowl-edged" appears.

Table A–6. The netmain Alter Logging Controls Submenu (Cont.)

Task	Command Box Used	Explanation
Get assistance with the menu	F7—Help me use the menu	Gives online help for using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Logging Controls Submenu

The number of option boxes that you see in the Alter Logging Controls submenu depends on the monitor's current logging behavior. If you start **netmain_srvr** with the default option, log files are always written (unless you stop them), and the submenu displays three option boxes that describe the log file parameters. If the **-nl** option was used at monitor startup, no logging is performed (unless you start logging), and the submenu displays just one option box. Figure A–8 shows the top sections of both menus.

In both versions, the long rectangular left-hand boxes report the status of each parameter. The long rectangular right-hand boxes name or describe the parameter. To change a parameter, you can select either the left-hand or the right-hand box. To start logging, select either box in Menu A. Menu B then appears.

Use the first box in Menu B, "Logfile is already being written", to close a log file. The options at monitor startup determine whether a new log is written. If the **-nl** option was used, no new log file will be written. If the option was not used, a new log file starts once you close the original file. Before closing a file you plan to analyze, note its name; you must enter the name in the Analyze Network Data menu. Use the second box in Menu B, "Original log file name", to change the name of a log file. **Netmain** automatically closes the current file and opens a new log file with the filename you provide or with the default **name net_log.yy.mm.dd**. (Either name is relative to the **'node_data/system_logs/net_log** directory.) If **netmain** creates more than one default log file on the same date, it appends a suffix, using alphabetical order, to the log file names. Three log files created on the same day would be named

net_log.83.08.19 net_log.83.08.19_a net_log.83.08.19_b

When you change the name, the new name appears in the left-hand box, and the message "Logging will be shut down and restarted" appears. If you choose to use the default name, the next message says "Using default name."

Use the bottom box in Menu B, "Limit on log file length", to display the length to which the current log file can grow or to limit the length of the new log file (the one displayed in the middle box). The length is in units of 1 kilobyte (1024 bytes); the default length is 3000 (3 megabytes). You cannot change the limit for a current log file.

Note that if a log file reaches the maximum length and **netmain** needs to write more data, the program starts to write over the oldest data in the log file. If you want to save this oldest data, you should close the file and store it when it approaches its maximum length.

If you attempt to leave the submenu without choosing "F1–Perform logging changes" or "F2–Revert to original logging," an input box prompts "Do you want to perform the changes? [Y | N]." You must choose to perform the change or cancel the change in order to leave the menu.

The netmain Alter Probe Timing Submenu

The Alter Probe Timing submenu shown in (Figure A–9) displays the schedule used by each probe when it collects data. These parameters are the same ones as the following **netmain_srvr** options:

–sample	Probe time
–skip	Probe distance

Use the Alter Probe Timing submenu to

- Display the intervals at which probes collect data.
- Change the scheduling intervals. Probes and their default data collection intervals are listed in Section A.4.

Table A–7 lists each command box in the Alter Probe Timing submenu.

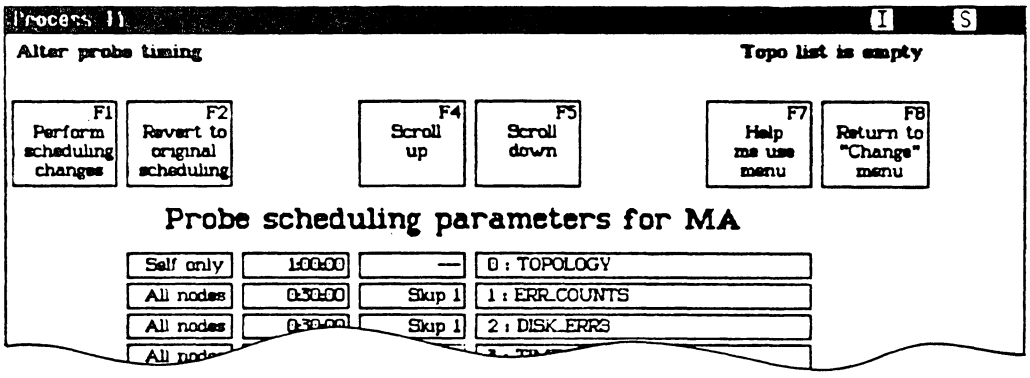


Figure A–9. Alter Probe Timing Submenu

Table A-7. The netmain Alter Probe Timing Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform scheduling changes	Causes the changes you make to take effect.
Cancel changes to parameters	F2—Revert to original logging	Causes changes made to return last scheduled intervals in effect. When you use the Alter Probe Timing submenu, you're making changes to parameters on a "scratch pad." Select F1 to cause the changes to take effect. If you make changes, then decide not to put them in effect, use F2 before you use F1 to undo the changes in the scratch pad. The changes you make with F1 take effect when the status message "Node_name acknowledged" appears. If you make further changes to parameters and choose F2, the logging parameters revert to the parameter changes that were most recently in effect.
Show probes above those currently displayed	F4—Scroll Up	The menu displays only eight probes at a time. If you are in the lower portion of the menu, use F4 to display the top portion of the menu.
Show probes below those currently displayed	F5—Scroll Down	The menu displays only eight probes at a time. If you are in the upper portion of the menu, use F5 to display the bottom portion of the menu.
Get assistance with the menu	F7—Help me use the menu	Gives online help for using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Probe Timing Submenu

As shown in Figure A–9, which shows the top [art of the submenu, each probe name appears in the long rectangular box at the right; parameters for the probe are on the left. Table A–8 lists the probe parameters appearing in the leftmost boxes of the Alter Probe Timing submenu.

Table A–8. Probe Parameters

Task	Parameter Box Used	Explanation
Probe scope	First box on left	Indicates whether the probe operates on all nodes or only on the node on which the monitor is running. You cannot change this parameter.
Sampling interval	Second box from left	Indicates how often the probe operates in hh:mm:ss format. The actual probe sampling intervals are slightly longer than the number shown, especially on heavily used nodes. For information about probe defaults, see Section A.4.
Skip Distance	Third box from left	Indicates the distance between nodes checked in each pass around the ring. It is only relevant for probes that check every node. A default skip distance of 1 checks each node on each pass; a skip distance greater than the number of nodes in the ring checks a different node on each pass. For skip distances greater than 1, the probe starts with a different node on each pass. For example, a skip distance of 5 checks the following nodes on each pass:

Table A–8. Probe Parameters (Cont.)

Task	Command Box Used	Explanation
		Nodes 1, 6, ... on the first pass Nodes 2, 7, ... on the second pass Nodes 3, 8, ... on the third pass . . . Nodes 1, 6, ... on the sixth pass Note that the numbers in the above example represent distances from the monitor around the ring. Node 1 runs the monitor, node 2 is the next node downstream, and so on.

If you attempt to leave the submenu without choosing "F1–Perform Scheduling Changes" or "F2–Revert to original scheduling," an input box prompts "Do you want to perform the changes?" [Y | N]; you must choose to perform the change or cancel the change in order to leave the menu.

Guidelines for Scheduling Probes

When you first start to run monitors, you should schedule their probes to collect data very frequently so that you can compile a complete set of data. After several days or a week, you can reschedule the probes.

When you start **netmain_srvr** on a node, leave the TOPOLOGY probe at its default time of 1 hour (01:00:00). Within 48 hours, the log files should include a fairly complete topology list; every node will most likely have been powered on and be online and communicating on the network when the **lcnode**s were run. The **netmain** process then automatically reschedules the topology probe to run every 12 hours to conserve CPU time.

If you change the network topology by adding, removing, or relocating nodes, change the sampling interval to every half hour or so. The **netmain** process will activate the TOPOLOGY probe within 1/2 hour, and the **lcnode** performed by the topology probe will collect the information about new nodes. After this, the topology probe will again run only once every 12 hours.

Note that when a node does not appear in an **lcnode**, the **netmain_srvr** leaves it on the total node list for some time, assuming that it is powered down, that the node's loop is switched out, or that the node is not communicating on the network. If the node disap-

pears for more than a week, the **netmain_srvr** assumes that it is really gone and removes its name from the Total Node list.

Probes that can check each node write a large amount of data to log files. When you configure these probes for each node running a monitor, determine how much disk space you want to allow the log files to consume each day. In general, log file storage requirements

- Increase with the number of probes you use
- Decrease as the sampling interval increases
- Decrease as the skip distance increases

When scheduling probes for your network, remember that log files reach their maximum size most quickly when you check many nodes, sample frequently, and run many different kinds of probes. Sampling less frequently does not fill up the log file quickly, but does not get as much detailed information. If you run fewer probes, you do not fill up the file as fast, but you get less complete information about the network. On any schedule, the longer the recording period, the fuller the log file gets. Make trade–offs based on your needs for network information.

The **check interval** for each probe is the product of the probe's sampling interval and the skip distance. The check interval determines how often the probe on a particular monitor actually checks each node. The following equation shows the check interval.

c = rn s

Where:

c Is the check interval (i.e., how often the probe checks each node)

rn Is the probe n's sampling interval

s Is the skip distance

Using this formula, we could schedule a monitor's ERR_COUNTS probe, for example, to a skip distance of 10 and a sampling interval of 3 minutes (00:03:00). The check interval is 30 minutes, so the ERR_COUNTS probe checks each node once every 30 minutes.

If you are experiencing an ongoing problem with node or network performance or reliability, you should run probes at regular intervals, with more frequent samples and lower skip rates, in order to locate the problem. The smaller the check interval, the more data you will collect and the better your chances for detecting the problem.

When you change probe scheduling parameters, each probe wakes up as soon as it is rescheduled. If you reschedule a number of probes at the same time, avoid "bursts" of CPU activity by setting parameters for a single probe, (use "F1–Perform scheduling changes" to

change the parameters) and then setting parameters for the next probe. If you change all parameters at once and then perform the scheduling changes, the probes all wake up at once. For a brief period, these multiple wake-ups can affect performance on the node running the monitor.

The netmain Alter Observer Timing Submenu

The Alter Observer Timing submenu (shown in Figure A-10) displays the schedule used by each observer when it collects data. These parameters are the same ones as the following **netmain_srvr** options:

-observe Observer time

-reobserve Observer time

Use the Alter Observer Timing submenu to

- Display the intervals at which observers collect data
- Change the scheduling intervals

Observers and their default data collection intervals are listed in Section A.4.

Table A-9 lists each command box in the Alter Observer Timing submenu.

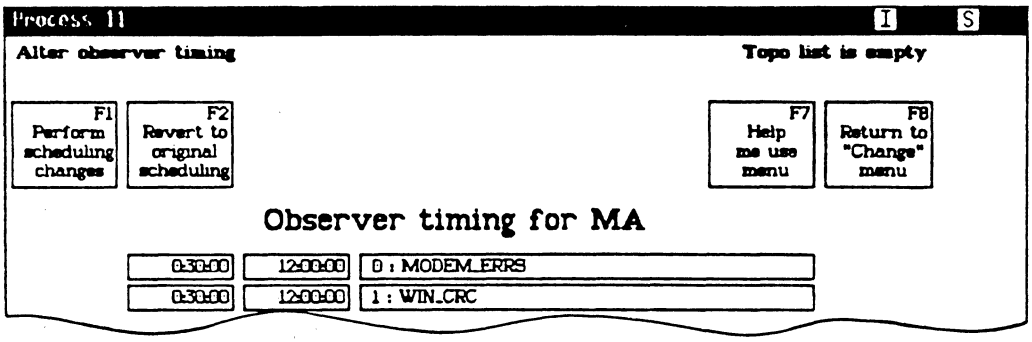


Figure A-10. Alter Observer Timing Submenu

Table A-9. The netmain Alter Observer Timing Submenu

Task	Command Box Used	Explanation
Change parameters of log files	F1—Perform scheduling changes	Causes the changes you make to take effect.
Cancel changes to parameters	F2—Revert to original scheduling	Causes changes made to return to the last scheduled intervals in effect. When you use the Alter Observer Timing submenu, you're making changes to parameters on a "scratch pad." Select F1 to cause the changes to take effect. If you make changes, then decide not to put them in effect, use F2 *before* you use F1 to undo the changes in the scratch pad. The changes you make with F1 take effect when the status message "Node_name acknowledged" appears. If you make further changes to parameters, then choose F2, the logging parameters revert to the parameter changes that were most recently in effect.
Get assistance with the menu	F7—Help me use the menu	Gives online help using the menu.
Exit this menu	F8—Return to "Change" menu	Returns to the Change Monitor Behavior menu.

Using the Alter Observer Timing Submenu

If you attempt to leave the submenu without choosing "Perform Scheduling Changes" or "Revert to original scheduling," an input box prompts "Do you want to perform the changes?" [Y | N]. You must choose to perform the change or cancel the change in order to leave the menu.

As shown in Figure A-10, the rightmost rectangular box on the menu names the observer, the middle box displays the observer's recheck interval and the leftmost box displays the observer's wake-up interval. The wake-up interval (30 minutes by default) is the interval at which the observer checks the data it has collected. When an observer reports on a node, it does not check data for that node again until after the recheck interval (12 hours

by default). A long recheck interval prevents numerous notifications about the same problem.

A.5.6 The netmain Analyze Network Data Menu

Use the Analyze Network Data menu to analyze the information **netmain_srvr** monitors collect about your network. This menu enables you to

- Specify the source of the data you wish to analyze
- Choose a data output format and appropriate parameters for that format
- View the data output format on the display monitor or save the display for printing at a later time

Figure A–11 shows the Analyze Network menu, and Table A–10 lists each command box in the menu.

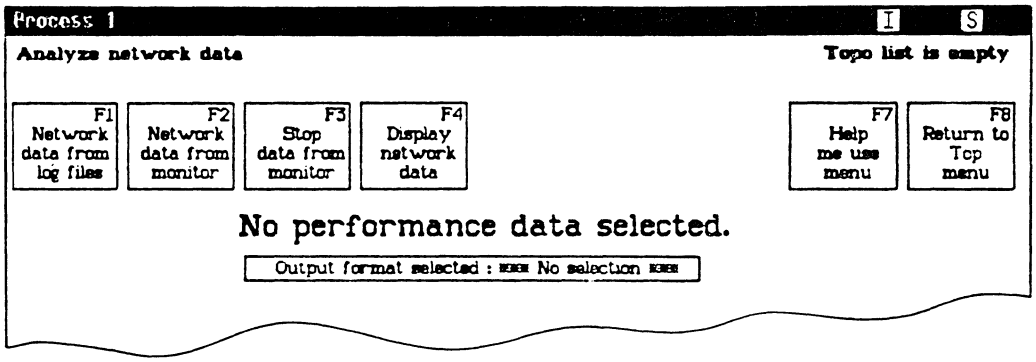


Figure A–11. Analyze Network Data Menu

Table A–10. The netmain Analyze Network Data Menu

Task	Command Box Used	Explanation
Select a log file	F1—Network Data	Brings up a submenu of log files you can analyze. If multiple log files are specified, **netmain** merges the contents of the files before performing the analysis.
Use data from the current monitor	F2—Network Data from monitor	You must select a monitor with F1 in the Find Monitors and Nodes menu before using this command. Data comes from a running monitor. To get enough data for analysis, use F2 on the Change Monitor Behavior menu to set probe check intervals to short time periods; e.g., 20 seconds. Set the time axis marks to about the same short intervals.
Stop data from monitor	F3—Stop data from monitor	Closes the data stream from a running monitor so it can be examined from **netmain**. After you issue the command, the monitor continues to collect data but does not send data to **netmain**.
Choose a display format output	F4—Display network data	Formats the network data by using the output formats you select. You cannot display data unless selection flags are replaced by some performance statistic or output format. For most displays, you should also choose a topology list, a monitor, and a log file. Data is displayed in a window that can be saved for printing with the DM commands **pn** or **pw**. CTRL/Q halts a display.
Get assistance with the menu	F7—Help me use the menu	Gives online help using the menu.
Exit this menu	F8—Return to Top–Level menu	Returns to Top–Level menu.

Using the Analyze Network Data Menu

Before you can use the Analyze Network Data menu to look at data, you need

- A source of data. Use one or more log files or a running monitor selected with F1 in the Find Monitors and Nodes menu.
- A Topo list. You can use one you've already generated with the Find Monitors and Nodes menu; or, if you don't specify a Topo list, **netmain** will generate one to build a display.
- An Output Format selection.

After you choose a source of data, the Analyze Network Data menu names the log file(s) selected or displays the name of the monitor you are examining. If the log file's pathname is long, **netmain** truncates it.

After you have a Topo list and a source of data, select an output format. **Netmain** displays additional menu items containing parameters for that format. Many parameter menu items have default selections, others display a "**No Selection**" message. When you make a selection or change an existing selection, additional menus or prompt boxes can appear.

After you supply the information necessary, **netmain** collects the data and generates the format you selected.

The following subsections describe what happens when you choose various command boxes from the Analyze Network Data menu.

Selecting the Log Files Submenu

Choosing F1 Select Log Files produces a menu that lets you analyze one or more of the log files you selected with the Find Monitors and Nodes menu. If you choose to analyze multiple log files, **netmain** merges the data from the files.

Merging multiple log files provides more complete information than using a single log file. Several log files may contain data from sequential time periods, for example. Also, if certain loops are switched out of the network and there are monitors in each loop, merging log files merges the data collected by the monitors on each loop. If you didn't merge log files, you simply would not see data for any loops that were switched out.

To select each log file, select one of the long, rectangular boxes in the menu and enter the log file's pathname in the input box that appears. The help information that appears lists the command search rules **netmain** uses if you specify a relative pathname.

If you choose a log file that you haven't closed, an error message appears. Close the file via the Change Monitor Behavior menu or select a working monitor instead.

When you analyze more than one log file, enter the log files in any order. If the log files were produced by monitors on different nodes, correct any differences between the Universal Coordinated Time (UCT) on the nodes running each monitor. If you don't correct these differences, or correct them imprecisely, displays will show numerous "x" marks, representing loss of data.

UCT differences usually exist only in networks that have multiple monitors. Log files copied from one node to another, however, retain the UCT of the node on which they were created.

If you have let the TIME_SKEW probe run, simply select the "F1 Compute offset times" command to make **netmain** correct the time differences by using the TIME_SKEW data. If you use the default sampling interval for this probe, the probe will probably have adequate data to perform the automatic computations. (If it does not have adequate data, the computation will fail.) Using the F1 command with the TIME_SKEW probe is the best offset time correction method.

If you have not let the TIME_SKEW probe run, enter the offset times yourself by using the small box at the right of the box that contains the log file name. Find out the UCT time on each node from which you've chosen a log file. Go to each node, use the **date** shell command, and record the UCT displayed. Be very exact, recording the time it takes to walk between nodes and subtracting it from the UCTs you gather. Then, with the times written down, choose one node as the node with the "base" UCT. Compute the difference (offset) between the base UCT and the UCT on any other nodes from which you've chosen log files. Enter this offset in the form "+hh:mm:ss" by choosing the offset time box and using the input box that appears.

You can also use the **lcnode** command to find the times on nodes that run monitors. If there are many nodes in the network, take into account the amount of time it takes to product the **lcnode** display.

If you use the **calender** utility to reset the UCT on a node running a monitor, there could be an offset between log files created before and after you set the UCT. (Note that time zone settings do not affect the UCT.) To correct for this offset, manually enter the offset between the two times, as just described. Use the old time as a base time, and, for any log file created after the calendar reset, enter the new time as an offset of the old time.

Selecting the Executing Monitors Submenu

Before selecting an executing monitor as the source of data for a **netmain** display, reschedule the probes you want to examine to operate more frequently than the defaults. If you do not reschedule probes, **netmain** cannot produce the display until all nodes have been sampled at least twice. This can take a very long time with default probe scheduling.

A monitor that samples frequently can produce a large amount of data. Ensure that it does not reach its maximum size and begin to write over old data. To provide prompt display of data, take the following steps:

1. Use the Change Monitor Behavior menu to close the current log file. Start a new log file to record data while you examine the active monitor.

2. Reschedule the probe(s) you want to examine to check each node every 15 to 60 seconds.

3. Use the Analyze Network Data menu to select data from the current monitor.

4. From the same menu, set the time axis marks equal to the node check interval. If the node check interval is 00:00:30, set the time axis marks to 00:00:30.

5. Use F4 Display Network Data to produce a real-time display of data from the executing monitor.

6. Once you've finished examining the data from the executing monitor, type CTRL/Q or CTRL/N to stop output to the display.

7. Use "F3 Stop Data from Monitor", and return to the Change Monitor Behavior menu to reschedule the probes to their original sampling schedule. You can start another log file.

A.5.7 Choosing Output Formats for Data

This subsection describes the output formats available for analysis, the performance statistics or other data available in each format, and the parameters to specify for each format. It also provides additional notes about interpreting output in various formats.

Netmain provides graphic display output formats, as well as printed and binary data files. You can view some kinds of data in several different output formats.

Choose a format by selecting "Output format selected" in the Analyze Network Data menu.

- Choose a "density plot" format (gray scale plot) to get a graphic display that shows the behavior of all nodes.

- Choose a "peaks" format (scatter plot) to isolate nodes whose behavior on a performance statistic is above a threshold level.

- Use an "across net" format to survey nodes' relative contributions to network-wide counts for a performance statistic.

- Use the "bar chart" format to look at only a single node's behavior, or to see exact percentages and counts for a performance statistic.

Output Format Descriptions

Table A–11 lists each of the formats and provides a brief description of each. Use the first three formats listed in the table to get information about diskless nodes and network events, for example, hardware failure messages.

Use the other formats listed in the table to display performance statistics collected by **netmain** probes. Performance statistics for the entire network and for individual nodes are available in a number of categories. When you select an output format, a menu of parameters for that format appears. Use the parameter selection menu to choose a category and a performance statistic within that category. (The next subsection, "Output Format Parameters" and Table A–12 describe the parameter menus.)

In most displays, information about a performance statistic appears as a percentage of a total for that category. For example, choosing a performance statistic in the ring transmit category shows occurrences of each node's ring transmit statistics as a percentage of the node's total ring transmits.

"Rate" and "Across net" formats do not show percentages of node totals in a performance statistic category. "Rate" formats, such as "Rate density plot" or "Peak rate" show the absolute number of counts per time unit. In "Across net" formats, such as "Density across net" or "Peaks across net," the percentage shown is each node's contribution to network totals for that statistic.

In the graphs, time intervals appear on the vertical axis. Most of the graphs show data for all nodes in the topology list and show the nodes on the horizontal axis. If the topology list is empty, **netmain** performs an **lcnode** and uses the results.

Table A–11. Output Format Descriptions

Format	Description
Diskless partners	Displays the network's diskless nodes and their partners. Configure the display to show the diskless nodes served by each "mother" disked node, or to show the mother node for each diskless node. Use this format only when you analyze data from log files not from executing monitors.
Scattergram events	Makes a scatter plot of incidences of probe failures or hardware error messages. Hash marks represent event incidences higher than a threshold level set in the parameter menu for this format.

Table A–11. Output Format Descriptions (Cont.)

Format	Description
Bar chart	Makes a bar chart for the performance statistic you specify in the parameter selection menu. Each hash mark (#) in the chart represents incidences of the condition tracked by the statistic, as a percentage of a total for that statistic. For example, a bar chart for a performance statistic in the ring transmit category shows percentages of total ring transmits. Unlike other output formats, you can display a performance statistic for a single node in a bar chart. For a single node, the bar chart shows the statistic as a percentage of a total for that node. If you choose "aggregate," the bar chart shows the statistic as a percentage of a total for the entire network. The bar chart is also the only format that provides exact percentage levels and quantities for the performance statistic chosen.
Peak incidences	Makes a scatter plot of the "peak" incidences of the performance statistic chosen. Hash marks represent a percentage of incidences higher than a threshold you set for the "Top of error scale" parameter in the parameter menu for this format.
Incidence density	Makes a gray scale plot that graphically displays the percentage level for the performance statistic chosen. The percentage level for each node is proportional to the density of the plot. **No color** represents 0%, or a percentage too low to report at the resolution set and a **solid color** represents percentages over the "Top of error scale" number set in the parameter menu. The name for this condition is **saturation.**
Peak rate	Makes a scatter plot that shows nodes that exceed a specific threshold rate (occurrence per time interval) for the statistic chosen. Hash marks represent a rate higher than the threshold set with the "Top of rate scale" parameter in the parameter menu for this format.
Rate density plot	Makes a gray scale plot that shows, for each node, the rate of occurrence of a condition measured by the performance statistic you select. The rate for each node is proportional to the density of the plot. No color represents a rate too low to report at the resolution set, and a solid color represents rates over the "Top of rate scale" set in the parameter menu for this format. The name for this condition is **saturation.**
Peaks across net	Makes a scatter plot that isolates nodes that contribute the highest percentages to network totals for the performance statistic you select. Hash marks indicate nodes whose contribution is higher than the threshold you set with the "Top of error scale" parameter.

Table A–11. Output Format Descriptions (Cont.)

Format	Description
Density across net	Makes a gray scale plot that shows the relative contribution of each node to network–wide totals for the performance statistic chosen. The percentage contribution for each node is proportional to the density of the plot. No color represents a contribution too low to report at the resolution set, and a solid color represents rates over the "Top of error scale" set in the parameter menu for this format.
Printed output	Displays a printed summary of events. You can choose to summarize incidences of hardware failure messages, user messages, probe failures, observer reports, or memory errors.
Binary data file	Dumps data into a binary data file that you can use in specialized network monitoring programs you may wish to write. This format is for advanced users who want statistics not provided by **netmain**. See the insert file **/sys/ins/nud.ins.pas** for the binary file record structure, and see the files **/domain_examples/netmain/nud_example.pas** or **/domain_examples/netmain/nud_example.ftn** for an example of how to use it.

Output Format Parameters

When you select an output format, a menu of appropriate parameters for that format appears. The menu items are identical for many output formats. Table A–12 describes each parameter and shows the output formats for which it is required.

Table A–12. Parameters for Output Formats

Parameter	Required By These Output Formats	Parameter Function
Select a log file	F1—Network data from log files on the Analyze Network Data menu	Brings up a menu of log files that you can analyze. If multiple log files are specified, **netmain** merges the contents of the files before performing the analysis.

Table A–12. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
Earliest/Latest data displayed	All, when you use these files for analysis	Lets you limit the time range over which errors are reported, when you're using log files for analysis instead of a monitor.
	None, when you use a monitor	By default, reports start with the earliest data "First in log" and end with the latest "Last in log." To shorten the time range, enter a later time in the prompt box when you choose earliest data displayed, or enter an earlier time in the prompt box when you choose latest data displayed.
Time axis marks	All, except "Printed output" and "Diskless partners"	Lets you change the length of the time units used in formatted data outputs.
Max wait for data	Same as above; also not necessary when you select "bar chart" with actual sampling times	Sets the number of time axis intervals that **netmain** will wait for data from a missing node. After the specified time, **netmain** continues to format data. When you are analyzing data from a log file, you can enter "end–of–file" to read to the end of a log file before continuing with the scattergram. You cannot choose "end–of–file" when you are examining an executing monitor.
Top of error scale	"Bar chart" "Density across net "	Sets the full–scale value or the threshold used in the output format you've chosen. In gray scale plots, incidences higher than the value set cause saturation, so the value you set is a full–scale value. In scatter plots, incidences higher than the value set show as hash marks, so the value you set is a threshold.

Table A–12. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
		The value set is given in percent units or, for rate plots, as a ratio of counts per time unit). The meaning of the percent unit depends on the format. For output formats that only show levels for performance statistics, the value is percentage of the total that results this condition. For "across net" formats, the value is the percent that each node contributed to the total number of errors of that type over the time interval. The default value is 50%.
Data plotted	"Bar chart" "Density across net "	Specifies the data value you see in plots and scattergrams. Displays a submenu (or peaks) from which you choose one of the categories shown in Table A–11. See Section A.4 for information about specific performance statistics.
Data dumped	"Binary data file"	Specifies data as in "data plotted," above. Here, however, **netmain** sends data to a binary file instead of formatting it for screen display.
Events plotted	"Scattergram events"	Lets you create a scattergram of hardware failure messages or probe failures. See Section A.4 for information about these events.
Data printed	"Printed output"	Lets you choose printed reports on hardware failure messages, probe failures, user messages, or memory errors.
Data ordering	"Diskless partners"	Lets you specify whether the "Diskless partners" format shows diskless nodes for each diskless node (choose "Partners by Diskless".)

Table A–12. Parameters for Output Formats (Cont.)

Parameter	Required By These Output Formats	Parameter Function
Node to plot for	"Bar chart"	Displays an input box that lets you specify the name of the node for which statistics are displayed in a bar chart. By default, the chart shows the sum for all nodes (network aggregate).

Interpreting Bar Chart Displays

The left section of a bar chart contains four columns of information about each time interval. The time and date that each interval ended are in the far left columns. The actual number counted for the performance statistic during this interval, and the number as a percentage of a total for the performance statistic, are in the next two columns. **Netmain** rounds off percentages to the nearest integer value. Each hash mark (#) represents a part of this percentage.

The actual percentage represented by one hash mark depends on on the scale you choose for the graph, using "Top of _____ scale." This value is the full–scale value of the bar graph. With a full–scale value of n percent, the bar graph is at its maximum or full–scale length (50 hash signs) when n percent of the transmit or receive I/O operations result in the error selected. If you use the default value of 50 percent, then the bar graph contains 50 hash signs when 50 percent of the operations result in the error selected. Using the default, then, each hash sign stands for a percentage of 1 percent.

If a node has been rebooted or if a probe has failed because of a transmit failure, the plot reports the reboot or transmit failure during the time interval in which it occurred. A set of greater–than (>) characters to the right of the hash marks indicates that the graph is off scale. Make the "Top of error percentage scale" higher if this problem occurs.

When you choose to plot for a single node, you should set the time axis intervals to correspond to the node's check intervals. This prevents inaccuracies due to the skewing between check intervals and time axis intervals.

When you interpret an aggregate plot, be aware of the limits on the plot's **resolution**. The node check intervals determine the resolution of the data. A check interval of 1 minute will of course produce higher resolution data than a check interval of 30 minutes, but you may be using a check interval of 30 minutes because of log file size limitations. Whatever your check interval, try to set the time axis interval in your aggregate plots to about the same number. Then, when you interpret the plots, note that displays may spread the error incidences out over a period of roughly two check intervals.

For example, if the check interval is 30 minutes and a 2–minute period of network problems occurs, many nodes may not be sampled until up to a half hour after the 2–minute period. Instead of showing a peak for a 2–minute period, the errors could be spread out over a 1–hour period.

If the time axis interval is much longer or much shorter than the check interval, the displays will be skewed even more and will pinpoint peak times for a particular performance statistic less accurately. If the check interval is much *longer* than the time axis interval, the times reported will be very inaccurate.

When a node is rebooted, it resets all its internal counters. The counts for a time axis interval during which a node was rebooted include only those statistics counted after the reset. The counters lose any quantities counted in the time axis interval before the reset. For this reason, time axis intervals which include node rebootings can report an artificially low incidence for the statistic. Charts for single nodes can show "node rebooted" messages in place of a report for that time interval.

Interpreting Scatter and Gray Scale Plot Displays

Scatter plots for hardware or probe failures illustrate incidences of each of these events with an *n* or asterisk (*), where *n* is an integer between 1 and 9, and the asterisk represents any integer greater than 9. Plots for performance statistics show any failure to get information from a node as an *x*.

Peak times shown in scatter plots have the same resolution limitations that they do in bar graphs (discussed in the previous subsection). Also, you should ignore the first line in a scatter plot since it usually shows an incomplete time axis interval.

If your network is very large, with more than 225 nodes, and you attempt to create a plot showing all nodes, the Display Manager (DM) cannot display the rightmost columns in the display for each time interval. To avoid this problem and display all nodes, take the following steps:

1. Take the total node list from the monitor and display it, using the Find Monitors and Nodes menu.

2. Use the DM to cut and paste the list into a new file.

3. Edit the new file to give it a format acceptable to the "Topo list from data file" command. See **netmain** help for details on this command.

4. Now cut and paste part of this new file into a separate file. You should now have two files, each containing a list of half the nodes on the total node list.

5. Invoke the "Topo list from data file" command and name the first file. Then do a plot.

6. Now invoke the "Topo list from data file" command again; this time name the second file. Now do another plot.

7. You now have plots in two windows, each containing half of the nodes. You can line the two windows up by using DM window control commands so that the two windows give the effect of one very large window.

Saving Output Displays

You will often want to save **netmain** displays for future reference. When it makes a display, **netmain** creates a window and a pad, writes the text into the pad using special fonts, and closes the pad. You can save the pad in the fonts used by using the **pn** (pad name) command. Position the cursor in the window, then type the following command:

pn *filename*

Filename must be a file on your node. When you close the window in which **netmain** produced the display, (with CTRL/Y), the *filename* will contain the display in the original fonts. If you just cut and paste to a file, the special fonts will not be used, and the text may be less easy to read.

To print one of these saved files, open the file using <READ> or <EDIT>. Then use the **cpscr** shell command to make a copy of the entire screen display and store it in a second file. Print the file with the **-plot** option. You can use the following command sequence:

cpscr *second_filename*; **prf** *second_filename* **-plot**

See the *SysV, BSD, or Aegis Command Reference* manual for more information about the **prf** and **cpscr** commands.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing BSD System Software*
Order No.: 010853-A00
Date of Publication: June, 1988

What type of user are you?

_____ System programmer; language _____
_____ Applications programmer; language _____
_____ System maintenance person
_____ System Administrator
_____ Student Programmer
_____ Other

_____ Manager/Professional
_____ Technical Professional
_____ Novice

How often do you use the Domain system?_____

What parts of the manual are especially useful for the job you are doing?_____

What additional information would you like the manual to include?_____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)_____

_____
Your Name Date

_____
Organization

_____
Street Address

_____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

FOLD

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Managing BSD System Software*
Order No.: 010853-A00
Date of Publication: June, 1988

What type of user are you?

_____ System programmer; language _____
_____ Applications programmer; language _____
_____ System maintenance person
_____ System Administrator
_____ Student Programmer
_____ Other
_____ Manager/Professional
_____ Technical Professional
_____ Novice

How often do you use the Domain system?_____

What parts of the manual are especially useful for the job you are doing?_____

What additional information would you like the manual to include?_____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)_____

_____
Your Name Date

_____
Organization

_____
Street Address

_____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

FOLD


0108853-A0U