System Administration for DOMAIN/IX BSD4.2

Order No. 009355
Revision 00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: December, 1986
Latest Printing:
Updated:

This document was produced using the Interleaf Workstation Publishing Software (WPS). Interleaf and WPS are trademarks of Interleaf, Inc.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

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

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

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

Preface

We've organized *System Administration for DOMAIN/IX BSD4.2* as follows:

Chapter 1 An introduction to system administration with DOMAIN/IX.

Chapter 2 "Maintaining Root Directories," including procedures to catalog nodes, and managing root directories with **ns_helper**.

Chapter 3 A comprehensive discussion of the network environment, including both AEGIS and DOMAIN/IX startup files.

Chapter 4 "Creating and Maintaining User Accounts". This contains a description and discussions of the registries, the DOMAIN/IX */etc/passwd* and */etc/group* and their use in the DOMAIN environment, as well as information on updating and replicating this information.

Chapter 5 This chapter discusses system and software security, the ACL system and DOMAIN/IX modes and how they interact.

Chapter 6 This chapter contains reference material on **ns_helper** and other servers that can run under DOMAIN/IX.

Chapter 7 Chapter 7 descirbes how to configure and maintain TCP/IP in a DOMAIN/IX *bsd4.2* environment.

Chapter 8 This chapter describes the *bsd4.2* line printer system and how to configure and manage it on a DOMAIN/IX system.

Chapter 9 Chapter 9 describes the **uucp** subsystem and how to configure and maintain it on a DOMAIN/IX *bsd4.2* system.

Chapter 10 Chapter 10 documents the **sendmail** process and how it operates with DOMAIN/IX.

Chapter 11 This chapter contains the *DOMAIN/IX Programmer's Reference* pages for the administrative commands associated with DOMAIN/IX *bsd4.2*.

This manual is intended for users who are familiar with DOMAIN/IX *bsd4.2* software and DOMAIN networks. We recommend that you read one of the following tutorial introductions if you are not already familiar with the UNIX® operating system.

- 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.

This document also assumes a basic familiarity with the DOMAIN/IX system. The best introduction to the DOMAIN/IX system is *Getting Started With Your DOMAIN/IX System* (Order No. 008017). This manual explains how to use the keyboard and display, read and edit text, and manipulate files. It also shows how to request DOMAIN system services using interactive commands.

Related Manuals

The *DOMAIN/IX User's Guide* (Order No. 005803, revision 01) is the first volume you should read. It explains how DOMAIN/IX works, and contains extensive material on the Bourne Shell, C Shell, and Mail.

The *DOMAIN/IX Support Tools Guide* (Order No. 009413) describes various DOMAIN/IX utilities (e.g., **awk**(1), **lex**(1), **yacc**(1)) that can help with developing and maintaining programs.

The *DOMAIN/IX Command Reference for System V* (Order No. 005798, revision 01) describes all the UNIX System V shell commands supported by the *sys5* version of DOMAIN/IX.

The *DOMAIN/IX Programmer's Reference for System V* (Order No. 005799, revision 01) describes all the UNIX System V system calls and library functions supported by the *sys5* version of DOMAIN/IX.

The *DOMAIN/IX Administrator's Reference for System V* (Order No. 009356) describes all the UNIX System V system administrator commands and provides detailed information on system registries and servers supported by the *sys5* version of DOMAIN/IX.

The *DOMAIN/IX Command Reference for BSD4.2* (Order No. 005800, revision 01) describes all the BSD4.2 UNIX shell commands supported by the *bsd4.2* version of DOMAIN/IX.

The *DOMAIN/IX Programmer's Reference for BSD4.2* (Order No. 005801, revision 01) describes all the BSD4.2 UNIX system calls and library functions supported by the *bsd4.2* version of DOMAIN/IX.

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.

The *DOMAIN System Command Reference* (Order No. 002547) gives information about using the DOMAIN system and describes the DOMAIN commands found in the */com* directory.

The two-volume *DOMAIN System Call Reference* (Volume I, Order No. 007196, revision 01; Volume II, Order No. 007194, revision 01) describes calls to operating system components that are accessible to user programs.

Documentation Conventions

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

bold We use bold type to emphasize keywords in text and command-line examples. A keyword can be:

> the name of an executable system object (command or shell script) and any options (switches, regular expressions, or real pathnames) that the command or shell script accepts. For example, **ls -la**, or **man ls**.
>
> the name of a callable function, including all syntactically required punctuation. For example, **open**(*path, flags, mode*).

> Any system object that has its own reference manual entry. For example, **passwd**(4).

We do not use bold type for general emphasis. In our ASCII help files, bold type looks the same as Roman type.

italics

We use italics to emphasize:

> Names or pathnames of system objects. For example, */etc/passwd* or */tmp*.
>
> Names we use as stand–ins for names and/or values that you must supply. For example, **cat** *filename*, "...prints *filename* on standard output..."

A command line like: **ls** *[options] [files]* indicates that **ls** is a command that can be followed by one or more options and an optional file or files. By extension, this font usage appears in command options and option arguments: **–n***number* means do this function *number* times.

We also use italics to indicate the title of a publication, such as the *DOMAIN/IX Command Reference Manual.* We do not use italic type for general emphasis. In our ASCII help files, italic type is underlined.

`pica`

Where possible, we use the constant–width pica font (or another "typewriter" style font) in code fragments, shell or DM scripts, and scripts for commands like **awk**(1) and **sed**(1). In our ASCII help files, pica type looks the same as Roman type.

name(1)

Where a filename or command name is followed by a number or number–letter pair in parentheses, that number indicates the section (and, if a letter is included, the subsection) of the reference manual set in which you can find reference information on the named command or file. For example, you can find reference information on the **lex**(1) command in Section 1 of the *DOMAIN/IX Command Reference Manual* and information on the */etc/passwd*(4) file in Section 4 of the DOMAIN/IX Programmer's Reference Manual .

[brackets]

We use brackets to delimit optional command line switches (options) and arguments. Brackets are also shell metacharacters that delimit a range or character class.

<angle brackets>

We enclose the name of a keyboard key in this type of brackets, for example, <ESC> or <AGAIN>. The < and > symbols are also shell metacharacters used for redirection of input or output.

^<KEY>

A control function that you execute by pressing the <CTRL> key and the named <KEY> at the same time. For example, ^<D> sends an End–Of–File.

<CTRL><KEY>

Same as ^<KEY>.

. . .

Horizontal ellipses indicate that the preceding item can be repeated an arbitrary number of times. For example, **troff** *file* ... means that you can say **troff** *file1 file2 file3,* and so on.

We use vertical ellipses to indicate that an irrelevant portion of text has been omitted from an example.

Note that, when we begin a sentence with the name of a filesystem object, we always capitalize the first letter of the name unless this would result in an ambiguity.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the *DOMAIN System Command Reference*. Refer to the **crucr** (CREATE_USER_CHANGE_REQUEST) shell command description. You can view the same description on-line by typing at the appropriate shell prompt:

```
/com/help crucr <RETURN>
```

For your documentation comments, we've included a Reader's Response form at the back of each manual.

Contents

Chapter 1 – System Administration Overview

Chapter 2 – Managing Root Directories

Chapter 3 – The Network Environment

Chapter 5 – Protecting System Registries and Software

Chapter 6 – Network and DOMAIN/IX Servers

Chapter 9 – Installing and Configuring uucp

Chapter 10 – Sendmail Configuration and Usage

Chapter 11 – Administrative Commands

List of Procedures

List of Tables

List of Figures

Chapter 1

System Administration Overview

This manual refers to all DOMAIN devices that communicate on the network as "nodes" in the network. A node can be a device with a display and keyboard, such as a DN3xx, DN4xx, DN6xx node, or a DOMAIN Server Processor (DSP) like the DSP80. Wherever appropriate, we distinguish between user nodes (with keyboards and displays) and DSPs (without keyboards or displays). For example, since you may use different methods to start a server on a node and on a DSP, we provide separate procedures for these two tasks. This chapter provides an overview of the software tasks generally involved in DOMAIN and DOMAIN/IX system administration.

Providing Network Services to System Users

As a system administrator, you are responsible for providing network services to system users. Although you define the exact scope of services for your site, this manual gives general guidelines for providing standard services. As a system administrator, you generally have the following responsibilities:

- Enable the network naming structure by cataloging nodes with disks, and providing services for diskless nodes.

- Create servers, which are processes that run on a node and provide access to some service, such as the use of a peripheral device. Chapter 3 describes how to set up servers, and also includes information about DOMAIN/IX startup files, environment variables, and the */etc* directory, where most of the DOMAIN/IX system services reside. Chapter 6 is a reference chapter on DOMAIN system servers and some of the DOMAIN/IX *bsd4.2* daemons; the rest of the Berkeley daemons are treated in Chapter 7.

- Create and maintain user accounts and a registry database. Chapter 4 provides instructions for creating and maintaining the registry, as well as information about generating and administering the DOMAIN/IX */etc/group*, */etc/passwd*, and the */etc/passwd.map* files.

- Protect the registry and system software, back up user files, and install new software on the network. Chapter 5 provides information about these tasks, as well as a discussion of DOMAIN/IX protection modes in the context of the DOMAIN Access Control List (ACL) protection system.

- Configure and administer DOMAIN/IX-specific services like **uucp**.

Managing the Network and Nodes

As a system administrator, you may also be responsible for maintaining the physical network, and for performing minor hardware upkeep procedures on nodes. (Your service representative performs all major hardware maintenance.) Complete information about maintaining the physical integrity of your DOMAIN network is available in the manual *Managing Your DOMAIN Network*.

In order to use the network management information in *Managing Your DOMAIN Network*, you should also be familiar with your own network's topology, the path travelled by information through the network. To understand your network's topology, you should know the direction of data flow in your network, the nodes' order of sequence in the data path and the locations of the cables that connect these nodes.

DOMAIN/IX System Administration

In addition to the system administration that the DOMAIN network itself requires, there are additional tasks that the system administrator must perform to make DOMAIN/IX run correctly on a network of machines. These are treated briefly here, and explained in full in later chapters.

In Chapter 3 of this book, you will find a complete discussion of the DOMAIN/IX environment, including the */etc* directories and their functions, the startup files specific to DOMAIN/IX and what you can do with them, and a description and explanation of pertinent DOMAIN/IX environment variables and how they affect the system.

Chapter 4 contains information about the */etc/passwd* and */etc/group* files, how to generate them, how to administer user accounts, and how DOMAIN/IX accounts relate to the DOMAIN network registries. In Chapter 5, we discuss the concept of super-user (root) in DOMAIN/IX, as well as the relationship between DOMAIN/IX protection modes and DOMAIN ACLs, and in Chapter 6, we provide a list and explanation of the various DOMAIN/IX servers and daemons, including **inetd**, the super-daemon.

> **Note:** In addition to the system administrator's command pages in Chapter 11 of this book, you will find references throughout *System Administration for DOMAIN/IX BSD4.2* to other sections of the *DOMAIN/IX Programmer's* and *Command Reference Manuals*. Of special interest to your work as a system administrator will be the pages in section 5 concerning file formats, e.g., **passwd**(5), and those in section 4 which deal with special files and certain kinds of hardware support, e.g., **inet**(4f). You should also be thoroughly familiar with the information in the *DOMAIN/IX User's Guide*.

Many of the system parameters that in a standard UNIX system must operate the same way system-wide, can be set differently for individual nodes in a DOMAIN/IX network. Discrete system startup files, for example, mean that each node can create a different system environment, depending on how you will use the node. DOMAIN/IX users thus have far more control over their individual working environments, which means a lesser burden for the system administrator. However, the system administrator is usually the primary source of information for the average user on how to tailor his or her DOMAIN/IX environment.

The advantage of DOMAIN/IX in a distributed environment is that it distributes the administrative workload between the administrator and the node's user. In this book, we attempt to describe those functions that are under the control of the system administrator. Those system parameters by which individual users can tailor their environments without affecting the operation of the network or DOMAIN/IX on the network are covered in the *DOMAIN/IX User's Guide* and *Getting Started with DOMAIN/IX.*

The Structure of DOMAIN/IX

The *bsd4.2* DOMAIN/IX system comprises three directories: */bin*, */etc*, and */usr*. These directories are analagous to the UNIX directories of the same names, and generally contain the same material. Each of the sections below lists briefly what you can expect to find in these directories.

/bin		Binaries, executable commands
/etc		Miscellaneous commands, some system administration commands, local commands, configuration files
/usr		Subsets of functions, generally of interest to end users. Contains:
	/usr/adm	Administrative directories
	/usr/dict	Directories for use with the **spell** command
	/usr/include	Include files for programs
	/usr/lib	Command and program libraries
	/usr/local	Local user–level commands
	/usr/man	On–line DOMAIN/IX Command and Programmers' Reference Manual pages
	/usr/pub	Various character sets
	/usr/ucb	Enhancements to UNIX originating at University of California, Berkeley
	/usr/spool	Line printer and **uucp** spooling and queuing directories

The DOMAIN/IX Administrator's Node

As a DOMAIN/IX system administrator, you should be aware of the fact that the system is structured to support two types of DOMAIN/IX nodes, a *system administrator* node and a *user* node. In addition, the installation procedures provided with SR9.5 include an installation procedure for creating *system administrator* nodes, and an installation procedure for creating *user* nodes. A *user* node contains those files and directories necessary to create shells and run commands and links to the system administrator node for essential system–level services.

A *system administrator*'s node will normally contain those one–of–a–kind components of the DOMAIN/IX system that are necessary for DOMAIN/IX to work properly on a DOMAIN network. For example, you need a set of spooling directories for the **lp** line printer subsystem, and another set for **uucp**, but you'll normally only have one set of each per network, even if your network consists of several DOMAIN networks joined in an internet. It is possible, however, for you to have more than one set.

At installation time, you may designate one node as the one 'master' administrator's node; you may have only one of these on your network, even if your network consists of more than one network. Also at installation time, you may set up one or more 'slave' administrator's nodes, which will replicate some of the important information on the master administrative node so that DOMAIN/IX will still run if the master node is down for some reason.

The *system administrator* installation procedure installs the following directories on a node that the *user* installation does not. There will be one of each of these directories on the master administrator's node on your DOMAIN network. Copies of directories that reside on the 'slave' administrator's node(s) will be updated automatically by means of shell scripts supplied and installed with the DOMAIN/IX product at SR9.5.

- The */etc* directory and its subdirectories.
- The */usr/spool* directory and its subdirectories.
- The */usr/lib/uucp* directory and subdirectories.
- The */usr/msgs* directory.

Probably the most important directory on the system administrator's node is the */etc* directory. As its name implies, it contains miscellaneous, but vital, components of the system, including system administration commands, communications daemons, configuration files, and the like. It is extremely important that there be either only one */etc* directory on a network, or if there are multiple */etc* directories, that they always be identical, for reasons explained below.

The DOMAIN system has one master registry which stores account information about **person, project,** and **organization uids** on your DOMAIN system; that is, it contains the details of your system's users' accounts. Each node has a local 'site' registry to use if the master registry is unavailable, but there is only one database that contains user account information like log-in names and passwords. Under DOMAIN/IX, this registry information is placed into the */etc/passwd* and */etc/group* files via the **crpasswd** command, and a file named */etc/passwd.map* is created to map DOMAIN person and project **uid**'s to DOMAIN/IX user and group **ID**'s and vice versa.

Thus, if you maintain more than one master registry or more than one */etc/passwd*, */etc/group*, or */etc/passwd.map* file on the same network or DOMAIN Internet, the DOMAIN/IX system may have difficulties in mapping the DOMAIN **uids** to DOMAIN/IX **IDs**, and vice versa. For example, if you had two */etc*'s, a DOMAIN/IX **user id** might be mapped to one DOMAIN **uid** in one *passwd.map* file, and to another **uid** in other; this could confuse DOMAIN/IX programs trying to use that information. These subjects are treated more fully later, but if you understand that only one master registry database and one master */etc* directory per network is allowed, you will minimize many problems.

DOMAIN/IX–Specific Accounts

Correct operation of DOMAIN/IX *bsd4.2* requires that your system have the following registry accounts created.

- daemon.%.%.%
- root.staff.%.%
- root.sys.%.%
- uucp.%.%.%
- %.daemon.%.%

The installation procedure instructs you to create all of these accounts. If you are updating existing software, rather than installing it for the first time, you may already have some or all of these accounts established. Use the DOMAIN **/com/edppo** and **/com/edacct** commands to create these accounts.

Chapter 2

Maintaining Root Directories

NOTE: This chapter applies mainly to installations with a single DOMAIN network. If you have a DOMAIN internet, that is, multiple DOMAIN networks connected by routing service, see *Managing DOMAIN Internets* for additional information about how to catalog nodes and maintain root directories.

Each node has a copy of the root directory that contains the names and hexadecimal IDs for nodes on your network. The root directory provides the associations between the node names and the corresponding ID numbers used by the network. To ensure file access and communications on the network, you must make sure that these directories remain accurate. This section describes the process of cataloging node name – node ID associations in root directories and of maintaining root directories, and gives step-by-step procedures to carry out these tasks.

Cataloging a Node

The **/com/ctnode** command catalogs a node. That is, it 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 previously cataloged node's name
- Replace a node's disk
- Run the **invol** utility
- Your service representative replaces the node's ID PROM. The 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 cataloged. We strongly recommend that you give names to all nodes in your network. Otherwise, you leave yourself and all users open to errors; hexadecimal numbers are difficult to remember and easy to type incorrectly. On networks that do not use the **ns_helper** (described below), remote nodes can only access nodes that have been cataloged.

Because each disked node has its own copy of the root directory, cataloging a node is a two-step process.

1. You must first catalog the node in its own root directory (or, for diskless nodes, in the partner's root directory).

2. You must then make this information available to all other root directories.

The procedures for step one are the same for all networks. The procedures for step two depend upon whether you run an **ns_helper** (naming server helper) process on your network. The **ns_helper** is a server process that maintains a master copy of the root directory and provides node name – node ID mapping information to nodes. It reduces the cataloging effort when you add nodes or change names. You must run the **ns_helper** if you have a DOMAIN internet, and you should run it if your network configuration changes frequently. However, you do not need to run the **ns_helper** on a small network where:

- Root directories usually are current

- Maintaining root directories takes little time

- New nodes are added to the network infrequently.

In this case, use the **/com/ctnode** and **/com/uctnode** commands to maintain the root directory.

The following section describes the procedures that catalog a node in its local root directory. The next section describes the procedures for maintaining the network's root directories using **ctnode**. The rest of the chapter describes how to use **ns_helper** on a DOMAIN network. See *Managing DOMAIN Internets* for information about using **ns_helper** on a DOMAIN internet.

Cataloging Nodes In Their Local Root directories

Procedures 2–1 and 2–2 catalog a node in its local root directory. Use these procedures for both disked and diskless nodes. 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 DSP server).

- 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, or 2–5 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 Customer Services representative.

PROCEDURE 2-1: Cataloging A Node

1. Log in as **user**.

2. Use the **/com/lcnode -me** command to determine the node's hexadecimal ID. For example:

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

3. Skip this step if this is a new diskless node, you changed the disk, or you ran **invol**.

   Use the **/com/uctnode** command to remove (uncatalog) the node's current name. 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.
```

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:

```
% ctnode new_name node_id -l
```

   For example:

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

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

```
% ctnode node_name node_id -l -r
```

5. Update the node's root directory with the names and ID's of all other nodes on the network:

```
% ctnode -update -l
3 nodes responded!
Node 8555 cataloged as "sam_node"
Node 8525 cataloged as "george_node"
Node 8523 cataloged as "raster"
```

END OF PROCEDURE 2-1

PROCEDURE 2-2: Cataloging A DOMAIN Server Processor

Use this procedure to catalog DSPs with disks. In new networks, catalog DSPs after you've cataloged nodes with monitors. Get the DSP's node ID from the white inspection slip attached to the shipping carton packing slip. If you do not have the inspection slip, contact your service representative, who will determine the node ID for you; 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.

1. Enter the following command at a node with a monitor to log in to the DSP as **user**:

   ```
   % crp -on node_specification -login user
   ```

   For example:

   ```
   % crp -on 8523 -login user
   Connected to node 8523
   ```

3. Skip this step if this is a new diskless node, you changed the disk, or you ran **invol**.

   Use the **/com/uctnode** command to remove (uncatalog) the node's current name. 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.
   ```

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:

     ```
     % ctnode new_name node_id -l
     ```

   For example:

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

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

     ```
     % ctnode node_name node_id -l -r
     ```

5. Update the node's root directory with the names and ID's of all other nodes on the network:

   ```
   % ctnode -update -l
   3 nodes responded!
   Node 8555 cataloged as "sam_node"
   Node 8525 cataloged as "george_node"
   Node 8523 cataloged as "raster"
   ```

END OF PROCEDURE 2-2

Using ctnode to Catalog Nodes on the Network

Once you catalog a node in its own root directory, you must then update the information in all other nodes' root directories, so that the remote nodes can access the newly cataloged node and its files. If the network is small and node configurations change infrequently, use the **/com/ctnode** and **/com/uctnode** commands to manage the network root directories. Procedures 2–3 through 2–6 describe the steps you must take to update the root directories. Use these procedures as follows:

- Use Procedure 2–1 or 2–2 for each node you are cataloging before using any of these procedures, unless the node's PROM was changed.
- Use Procedure 2–3 if you are creating a new network or adding several nodes to a network.
- Use Procedure 2–4 if you are adding a single node to an existing network,
- Use Procedure 2–5 if you are changing 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 replaces a node's PROM.

PROCEDURE 2–3: Creating a New Network

1. Log in as **user** at any node on the network.

2. Enter the **/com/ctnode –update** command to update the root directory to include 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<RETURN>
2 nodes responded!
Node 8555 cataloged as "sam_node"
Node 8525 cataloged as "george_node"
```

   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 Step 2 until you get a full root directory.

3. Update all other nodes with the information from this root directory. Enter the local node's name in the following command:

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

END OF PROCEDURE 2–3

PROCEDURE 2–4: Cataloging A New Node in an Existing Network

1. Log in to the new node as **user**.

2. Catalog the new node on all other nodes in the network; enter the node's name and ID in the following command:

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

END OF PROCEDURE 2–4

PROCEDURE 2-5: Changing Node Names

1. Repeat the following steps at each node on the network if you change a node's name. (Otherwise, the node will be cataloged under both the new and the old name.)

   a. Log in as **user**.

   b. Enter the **uctnode** command to remove the node's old name from the root directory. For example:

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

2. Log in to any node as **user**.

3. Update the root directories of all nodes in the network; enter the recataloged node's name and ID in the following command:

   ```
   % ctnode new_node_name node_ID -on //?*
   ```

END OF PROCEDURE 2-5

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.

1. Log in to any node as **user**.

2. Update 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 //?*
   ```

END OF PROCEDURE 2-6

Using ns_helper in Your Network

The **ns_helper** (**/sys/ns/ns_helper**), the Naming Server Helper, is a DOMAIN server process. It provides an automated method of maintaining node root directories. You can use **ns_helper** on any DOMAIN network, but you must use it on each network when two or more DOMAIN networks are joined in a DOMAIN internets. This section describes how to implement and use **ns_helper**.

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.

On large networks and on DOMAIN internets you can have more than one **ns_helper** process, each with its own 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	Consists of the network number (0 if you have only a single DOMAIN network) and the hexadecimal node_ID contained in a node's PROM.
Entry Type	The type of object being cataloged: for disked nodes, system directory (sdir); for diskless nodes, node.
UID	The Unique Identifier for a node's entry directory. If a node is diskless, **ns_helper** assigns a UID.
Entry Date/Time	The date and time at which this entry was made to the master root directory.
Creating Node	The hexadecimal ID of the **ns_helper** node at which this entry was made to the master root directory, i.e., the **ns_helper** that put this information in the database.
	Replica List
Replica List	The hexadecimal ID's of all **ns_helper** nodes. (See "Replicated Databases", below.

When to Use ns_helper in Your Network

Use **ns_helper** when:

- Node root directories get outdated rapidly and must often be updated using **ctnode -update**
- You want to maintain root directories from a single location
- Many nodes in the network have multiple users who are not responsible for node management and updates
- New nodes are added to the network regularly
- You have a DOMAIN internet. In this case, see *Managing a DOMAIN Internet*.

Replicated ns_helpers

In many networks, particularly in small ones, **ns_helper** running on a single node provides reliable access to network objects. In some environments, however, you should run the **ns_helper** server on more than one node. Consider running more than one **ns_helper** process when:

- There are many nodes in your network – a single server may not be adequate to handle the traffic
- You want to ensure the reliability and availability of the server process at all times — two or more servers can provide this insurance
- There are loops in your network that are switched out regularly and/or your network runs through several buildings -- you might want servers in each loop or building

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**. Therefore, the processes and databases are often called replicas. 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, or propagates, the new information to all the other nodes on the list. When **ns_helpers** receive new information from another server, they update their 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 of node, loop, or disk failure, 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.

Deciding on the Number and Placement of the ns_helper

Your network's topography and topology will influence the number and location of servers. Use the following statements as guides only.

- You must run at least one **ns_helper** server on each network on a DOMAIN internet.
- Run at least one **ns_helper** server for every 200 nodes in your network.
- If your network extends over several buildings, place at least one server process in each building.
- If there are loops in your network that are switched out regularly, place a server inside and outside the loop.

Managing Root Directories with ns_helper

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

When the naming server tries to locate an object, it first looks in the node's local cache of entry directory names. If the name is not in the node's cache, the naming server refers to **ns_helper**'s master root directory for information about the entry name. Also, if the naming server cannot locate an object using the locally cached information, it refers to the master root directory for more recent information about the name. Whenever the naming server gets information from the master root directory, it adds (updates) that information to the local node's cache.

When you use **ns_helper** on the network you do not need to use the **/com/ctnode** command to maintain a node's root directory except in the following unusual cases:

- When you change the name of a node (disked or diskless) in the **ns_helper** database, the old name is not deleted from the local caches. When you refer to the node by its new name the naming server updates the local cache. This means that both its old and new names are then in the cache. Use **uctnode** at each node on the network to remove the old entry from all local caches.
- If you interchange the names of two nodes in the **ns_helper** database, you must use **uctnode** locally at each node on the network to remove the old entries. When the naming server refers to the new name it updates the local cache with current information.

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

The edns Utility

The /com/edns utility enables you to manage the **ns_helper** and its database. Table 2–2 briefly describes the **edns** commands. The *DOMAIN System Command Reference* describes the **edns** commands in greater detail. On-line help is available by typing:

```
% /com/help edns
```

and

```
% /com/help edns commands
```

Table 2–2. 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 entries that appear in both, or that appear inconsistently in both.
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 the differences between two **ns_helper** databases, including both the master root directories and replica lists.
info	Displays the DOMAIN 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 the addresses all **ns_helper** nodes on the network, can display the nodes' 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** as a base.
quit	Ends the current **edns** session.
replace	Changes the address and UID associated with the specified name.
set	Sets the default **ns_helper** to be 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**'s 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 their 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 so that they refer to a single time standard. Check the node clocks periodically and reset them if they diverge by more than a few minutes. **Edns** provides a means of checking 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** at a time when network traffic is light and all nodes are connected to the network. At these times, most nodes can respond to requests for information about themselves from **ns_helper**.

edns and Diskless Nodes

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

diskless_$nnnnn

to a diskless node, where nnnnn represents the diskless nodes' 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 associates a UID that it generates for a diskless node with the name it gives to the diskless node. This information about the diskless node appears in the master root directory and can be copied to a node's local cache. If you use **ns_helper** in a single DOMAIN network, you should know that the UID and other information that **edns** generates for the diskless node serves as a place marker for information that is used in a DOMAIN internet; it has no effect in a single network.

As a general rule you should give diskless nodes non-default names. These names can be mnemonic and enable you to refer to the nodes in commands that take node specifications without using the node's hexadecimal ID. However, remember that the diskless node's name is not the name of an entry directory, and can not be used in a pathname argument.

To name a new diskless node on an existing network, simply use the **ctnode -root** command described in the next section, or use Procedure 2-10, below. If the node was on the network when the **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 **/com/ld // -ln** lists the names of all diskless nodes in the local copy of the root directory. For example:

```
% /com/ld -ln //
joes_diskless        diskless_$003476    nodisk              barebones
calamity             bridgel             comserver
```

The following commands tell you if a node is diskless.

```
% /com/netstat -n node_spec

% /com/lusr -n node_spec

% /com/bldt -n node_spec

% /com/lcnode -me
```

For example:

```
% /com/lusr -n //comserver
  b_jones          *** diskless //comserver ***  partner node: //bigdisk
```

User Procedures for Updating the Master Root Directory

The **/com/ctnode**, **/com/uctnode**, and **/com/ld** commands support a small subset of operations on the master root directory. On some secure networks where only the system administrator can use the **edns** command, any user can run these commands to manage the master root directory entries for a node.

- Use the following command to delete the entry for a node name in the local cache and the master root directory. If you remove a node from the network, you can use this command at any node to remove 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.

  ```
  % /com/uctnode node_name -root
  ```

  NOTE: Any time you change a node's name or interchange the names of two nodes, all users should also use **/com/uctnode** to delete the old entry or entries from their local root directory.

- Use the following command to add a node name in the local cache 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.

  ```
  % /com/ctnode node_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 local cache 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 reuse an existing name on a new node.

  ```
  % /com/ctnode node_name node_id -root -r
  ```

- You can list the contents of the master root directory by typing:

  ```
  % /com/ld -root
  ```

System Administrator Procedures for ns_helper

The rest of this chapter contains procedures for managing **ns_helper** processes and databases. Table 2-3 lists alphabetically the situations where you manage the **ns_helper** and its database and indicates the number of the procedure required to do the tasks. In secure networks, the ACLs might be set so that only the system administrator (%.%.sys_admin.%) can use the **edns** command. In this case, only the system administrator can use procedures 2-9 through 2-16. In this case, other users can run the **/com/ctnode** and **/com/uctnode** commands to manage their node's entry in the master root directory, as described in the preceding "User Procedures for Updating the Master Root Directory" section.

New at SR9.5 – uctnode and ns_helper

The **uctnode** command removes one or more specified entry directory names from the local copy of the network root directory. 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.

In networks running **ns_helper**, system administrators who suspect that the naming database on a node contains outdated information often use the **uctnode** command with wildcarding to uncatalog everything in the node's local network root directory, as follows:

```
% uctnode ?*
% ctnode this_node
```

This wholesale uncatalog forces the local naming server to get new information about nodes in the network from the master root directory and rebuild the local root directory with up-to-date information.

In earlier releases, the **uctnode** command verified the node name entry in a network root directory with the corresponding node in the network and removed the name from the root directory if the information was incorrect. At SR9.5, **uctnode** no longer queries nodes on the network before it removes names from a root directory; it simply removes the names.

The network root directory on each node is reserved for the entry directory names of other nodes, and should never contain references to other kinds of objects. The change in **uctnode** operation makes it important to adhere to this rule. If someone executes **uctnode** ?* on your network root directory, and your root directory contains a file or directory that refers to an object other than a node entry directory, **uctnode** will remove the reference to that object from the directory and you will no longer have access to that object by name.

Table 2-3. ns_helper Procedures

Purpose	**Procedure**
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 the synchronization of clocks on nodes running **ns_helper**	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, running **invol**	2-13 *

Note: An Asterisk (*) indicates a procedure that you can also do using the **ctnode** and **uctnode** commands, as described in the preceding "User Procedures for Updating the Master Root Directory" section.

PROCEDURE 2-7: Synchronizing Node Hardware Clocks

Use this procedure to correct node hardware clocks that are out of synchronization. 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.

1. Make sure you have an accurate timepiece as a standard time reference.

2. Put the node's NORMAL/SERVICE switch on SERVICE.

3. If the node has a display, use the DM **shut** command to 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. Follow the procedures for your model to shut down the node and display the mnemonic debugger prompt.

4. 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 1986/07/11 13:52:03 EDT.
   Would you like to reset it? Y

   Please enter today's date(year/month/day): 1986/07/11
   Please enter the local time in
     24 hr. format (hours minutes) 13:55

   The calendar has been set to 1986/07/11 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. Therefore, do not allow any objects to be created on the node for the length of time equal to the length of time by which you set the clock back. For example, if you set the clock back by one hour, you must wait one hour before you create objects on that node. You can accomplish this by waiting the setback period before you do step 5.

5. Return the service switch to the NORMAL position and reboot. On a node with a display, type:

   ```
   > RE<RETURN>
   > <RETURN>
   > EX AEGIS<RETURN>
   ```

6. Using the same timepiece you used in the previous step, repeat steps 2-5 at each of the nodes you selected to run **ns_helper**.

END OF PROCEDURE 2-7

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.

1. If more than one node runs (or will run) **ns_helpers**, check to make sure that the nodes' clocks are synchronized within one minute of each other. To do so, enter the **/com/netstat -n** command, followed by the node_specifications of the nodes that run **ns_helpers**. For example:

```
% /com/netstat -n //george_node //martha_node
     The net_id.node_ID of this node is 0.5678.
**** Node 4567 ****  //george_node
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 ****  //martha_node
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 reported times are not within one minute of each other, use Procedure 2-7 to synchronize the hardware clocks on the **ns_helper** nodes.

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

3. Edit the */sys/node_data/startup*[.type] script for the node type. Insert the line:

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

4. Start the **ns_helper** process.

- If you are working at the **ns_helper** node, 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.

```
% /com/crp -on node_specification -cps /sys/ns/ns_helper
```

5. Verify that the server process is running on the node; enter the **/com/pst** command. For example:

```
% /com/pst
---------------------------------------------------------
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
```

6. Repeat Steps 2 through 5 at any other nodes that will run **ns_helpers**.

END OF PROCEDURE 2-8

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: martha_node (ID 8521), and george_node (ID 8525), and with six nodes in the network: martha_node, george_node, fred_node, maxine_node, sam_node, and thomas_node.

1 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.

2. Use Procedure 2-8 to start **ns_helper** on each helper node.

3. Invoke **edns** from any node in the network. Select a node that will run **ns_helper**. Our example selects george_node; its **ns_helper** process becomes the default **ns_helper**. Enter:

```
% /com/edns node_id
```

The edns prompt, <edns>, appears. For example:

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

4. Use the edns **init** command to initialize the **ns_helper** database on the current default node. For example:

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

5. Enter the edns **ld** command to list the database and verify that it includes all nodes on the network. For example:

```
<edns> ld
fred_node     george_node    martha_node
maxine_node   sam_node       thomas_node
6 entries listed.
```

Repeat Step 3 if some nodes were not added to the directory. Skip to step 10 if you have only one **ns_helper** node on the network.

6. Use the following command to set the default server process to a replica **ns_helper** node:

```
<edns> set replica_node_id
```

For example, to set the default server to the process running on martha_node:

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

7. Use the following command to initialize the replica database from the first node.

```
<edns> init -from node_id
```

For example, to initialize the database from george_node:

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

8. Use the edns **diff** command to verify that the original and replica **ns_helper** databases are identical. For example:

```
<edns> diff martha_node george_node
The two directories are identical
The two replica lists are identical
```

   If the databases are not consistent, repeat Step 7.

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

10. End the **edns** session. Type:

```
<edns> quit
%
```

END OF PROCEDURE 2-9

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.

1. From any node, enter the **edns** command. For example:

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

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

```
<edns> add node_name node_id
```

For example:

```
<edns> add casey 7206
<edns> add filly 128c
<edns> add catherine_the_great 3333
<edns> add dr_paul 4b70
```

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

```
<edns> ld -n
```

For example:

```
<edns> ld -n

nodeid name
  7206 casey
  3333 catherine_the_great
  4B70 dr_paul
  128c filly
  503F fred_node
  8525 george_node
  8521 martha_node
  1C68 maxine_node
   5F6 sam_node
  70DE thomas_node

10 entries listed.
```

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

END OF PROCEDURE 2-10

PROCEDURE 2–11: Deleting Names from the Master Root Directory

Use this procedure if you remove a node from the network or to delete any entries that you added incorrectly to the master root directory.

1. From any node, enter the **edns** command. For example:

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

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

   ```
   <edns> del node_name
   ```

   For example:

   ```
   <edns> del casey
   <edns> del filly
   ```

3. Changes to the database take effect immediately. Use the edns **ld** command to list the directory to be certain you did not make errors.

   For example:

   ```
   <edns> ld
   catherine_the_great  dr_paul      fred_node
   george_node          martha_node  maxine_node
   sam_node             thomas_node

   8 entries listed.
   ```

END OF PROCEDURE 2–11

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 `catherine_the_great` to `cath_g`. 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.)

1. Repeat the following steps at each node on the network if you change a node's name. (Otherwise, the node will be cataloged under both the new and the old name.)

   a. Log in as **user**.

   b. Enter the **uctnode** command to remove the node's old name from the root directory. For example:

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

2. Log in to any node as **user**.

3. Enter the **edns** command. For example:

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

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

   ```
   <edns> del node_name
   ```

   For example:

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

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

   ```
   <edns> add node_name node_id
   ```

   For example:

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

END OF PROCEDURE 2-12

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 (see the DOMAIN System Command Reference Manual). These operations replace the node's entry directory UID with a new UID, and you must put the new information in the database.

1. From any node, enter the **edns** command. For example:

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

2. Use the following command to replace the information for the node:

```
<edns> rep node_name node_id
```

For example:

```
<edns> rep thomas_node 70de
```

END OF PROCEDURE 2-13

PROCEDURE 2-14: Adding or Initializing an ns_helper Replica

Use this procedure when you add an **ns_helper** replica. Use steps 2 through 5 to reinitialize the database of an existing **ns_helper** replica. In the following procedure's examples, we initialize the new **ns_helper** replica on node thomas_node, node_id 70de, from george_node, node_id 8525.

1. Use Procedure 2-8 to start **ns_helper** on the new node.

2. At any node, enter the following command to invoke **edns** and set it to the new replica node:

   ```
   % /com/edns replica_node_id
   ```

   For example:

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

3. Use the following edns command to initialize the replica database from an existing **ns_helper** replica:

   ```
   <edns> init -from replica_node_id
   ```

   For example:

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

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

   ```
   <edns> diff original_replica_node_name new_replica_node_name
   ```

   For example:

   ```
   <edns> diff thomas_node george_node
   The two directories are identical
   The two replica lists are identical
   ```

5. If the databases are not consistent, repeat Step 3.

END OF PROCEDURE 2-14

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 a **ns_helper** node with a slow clock being deleted and replaced by older (and inaccurate) data from a replica node with a fast clock. It is recommended that you keeep replica node clocks within one minute. **Ns_helper** allows more than one minute of divergence, and will indicate the skew only 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:

1. Invoke the **edns** utility. For example:

   ```
   % /com/edns
   The default ns_helper is 0.8525
   ```

2. 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
   ```

3. 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.

END OF PROCEDURE 2-15

PROCEDURE 2-16: Maintaining Consistency of Replicated Databases

Task 1: Comparing 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. To use the command enter:

```
<edns> diff node_specification1 node_specification2
```

For example:

```
<edns> diff martha_node george_node

                  value in          value in
                  0.008521          0.008525
diff             directory         directory          name

name            name found      name not found        maxine_node
 uid   21089D87.4000503f  2108af41.4000503f           fred_node

 The two replica lists are identical
```

In this example, **diff** shows that maxine_node is cataloged in martha_node's master root directory but not in george_node's. It also shows that fred_node 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 martha_node, george_node and thomas_node, the following two commands compare all replicas:

```
<edns> diff martha_node george_node

<edns> diff martha_node thomas_node
```

Task 2: Unifying Replica Databases

The **edns** utility provides two techniques 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 george_node's database any information that is present in martha_node's database but absent from george_node's. It also replaces any information in george_node's database that is older than information about the same entry in martha_node's database. It is important to note that nothing is changed in the "-from" replica (e.g., martha_node's) database. Type:

  ```
  <edns> merge george_node -from martha_node
  ```

  Since **merge** only operates on the target replica database, it is often appropriate to do a pairwise **merge** to bring two replicas into a consistent state. You would choose a pairwise merge if each database has some information that is missing from the other replica. For example, type:

  ```
  <edns> merge george_node -from martha_node<return>
  <edns> merge martha_node -from george_node<return>
  ```

- The **edns merge_all** command merges all replica databases, using either the defualt 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 procedure uses **merge_all** to ensure that all databases are consistent:

1. Invoke the **edns** utility. For example:

```
% /com/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. For example:

```
<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 step 2 is missing thomas_node (node_id 70de). Enter the following command to add the node

```
<edns> addrep thomas_node
```

4 Merge all replica databases; enter:

```
<edns> merge_all
```

In this case, the default replica node (martha_node) is the source node, and this command merges the information from george_node and thomas_node into martha_node. It then merges martha_node into george_node and thomas_node.

END OF PROCEDURE 2-16

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.

1. From any node, enter the **edns** command, for example:

   ```
   % /com/edns
   The default ns_helper. is 0.8525
   <edns>
   ```

2. Delete the replica; enter the **delrep** command. for example:

   ```
   <edns> delrep thomas_node
   ```

   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.

3. Check to see if the server process is still running from time to time. If you are at the replica node, enter:

   ```
   % /com/pst
   ```

   If you are at any other node, enter

   ```
   % /com/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** that has stopped, but the node is rebooted before you do step 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. For example:

   ```
   % edns 70de<RETURN>
   The default ns_helper is 0.70de
   <edns> info<RETURN>
   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. Enter:

   ```
   <edns> shut replica_node_id
   ```

   For example:

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

4. When the process has stopped, remove the line:

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

   from *'node_data/startup*[.type].

END OF PROCEDURE 2-17

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 if you are deleting this replica.

1. From any node, enter the **edns** command. For example:

```
% /com/edns
The default ns_helper is 0.8525
```

2. Shut down the replica **ns_helper**; enter:

```
<edns> shut replica_node_id
```

For example:

```
<edns> shut thomas_node
%
```

END OF PROCEDURE 2-18

Chapter 3

The Network Environment

This chapter describes the network environment of DOMAIN/IX. It provides information on:

- Node specifications
- The network naming structure, including information on such elements as entry directories and variant links
- The individual node's directory structure, including information on system software directories
- The DOMAIN/IX environment
- How to provide network services, including services on networks with both *sys5* and *bsd4.2* and with multiple administrative nodes
- Node activity at start-up and log-in, including information on start-up and log-in files
- How to administer diskless nodes

Node Specifications

Node specifications identify particular nodes on the network. Node specifications permit a node's communications software to locate other nodes. Typically, you use node specifications with DOMAIN shell commands that allow a **-n** node_spec option. The **/com/lusr**, **/com/netstat**, and **/com/lvolfs** commands are examples of DOMAIN shell commands that accept node specifications. A node specification can be either a hexadecimal **node_ID** or an ASCII **node name**.

Every node has a unique hexadecimal ID number, the node_ID, in a PROM (programmable read-only memory). You can specify any node in DOMAIN shell commands by its node_ID. However, since hexadecimal numbers are inconvenient for people to remember, you can also use a node name such as wonder_boy to specify a node.

You assign node names by **cataloging** the nodes. You catalog nodes with the shell command, **/com/ctnode** (catalog_node). Chapter 2 describes the procedures for cataloging and managing node names.

If your installation has two or more DOMAIN networks connected through routing services to make an **internet**, the nodes have a network number that is a prefix to the node_ID. The network number for a DOMAIN network that is not joined to other DOMAIN networks is, by default, "0". If your site has only a single DOMAIN network, you need not be concerned with network numbers and internet addresses. Simply use the node_ID with Shell commands that take the **-n** option. *Managing DOMAIN Internets* explains how you specify nodes in a DOMAIN internet.

Node Names: Disked Nodes and Diskless Nodes

You can assign node names to both disked and diskless nodes. On disked nodes the node name serves a dual purpose; it identifies both the physical node and the top level (node entry) directory on the node's naming (file) structure. (We describe the the naming structure in the next section.) Because diskless nodes do not have their own storage media, their node names only identify the nodes. They do not correspond to entry directory name and can not be used to access files. Instead, the pathnames to objects created by diskless nodes begin with the entry directory names of the nodes where the objects are stored.

As a result, you get an error message if you use a diskless node's node name with any command that takes pathnames as arguments. For example, if the node //casey is diskless, the **ls**(1) command:

```
% ls //casey
```

has no meaning and results in the message:

```
//casey not found
```

You can see the difference between the names of disked and diskless nodes in the output of the AEGIS **/com/ld** command

```
% /com/ld // -st
 Directory "//":

  sys
  type    name

  node    sprout
  node    radish
  sdir    snow_pea
```

The sys type (system type) of a diskless node name is "node." The system type of a disked node entry directory is "sdir" (system directory). "Administering Diskless Nodes" describes the procedures you use to provide partners for diskless nodes and to give them node name specifications.

The Network Naming Structure

The operating system expects information on nodes to be organized in a hierarchical tree-structure called the **naming tree**. Two naming tree components, directories (analogous to tree branches) and files (analogous to leaves on the branches), form text-string names called **pathnames**. You manipulate objects (directories and files) through their pathnames. Figure 3-1 shows the network naming tree, including standard software directories.

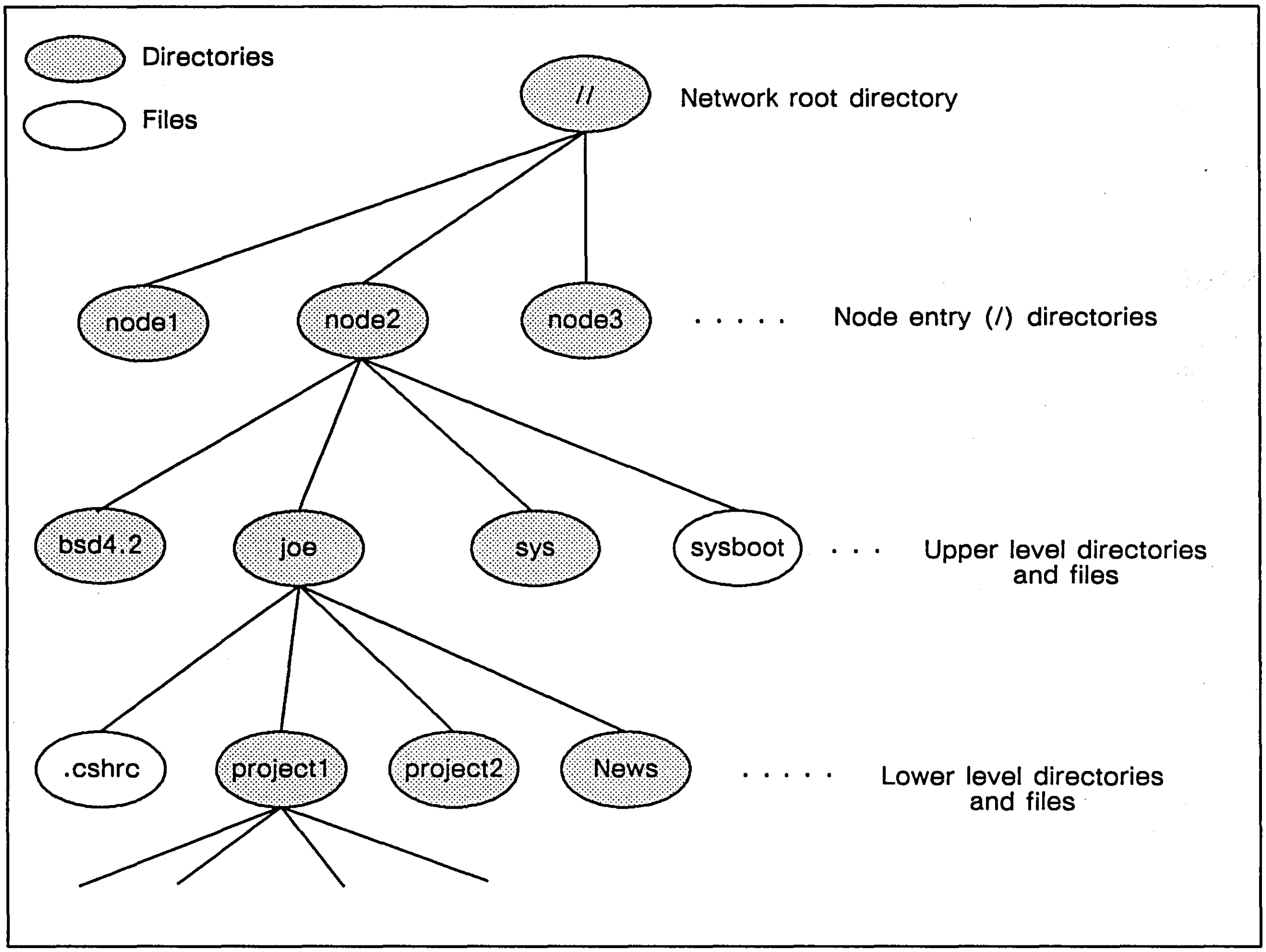

Directories
Files
//
Network root directory
node1
node2
node3
. . . . . Node entry (/) directories
bsd4.2
joe
sys
sysboot
. . . Upper level directories and files
.cshrc
project1
project2
News
. . . . . Lower level directories and files

Figure 3-1 Directory Structure

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 also the disked node's name, as described in the previous section.

The set of node entry directory names in your network is the **network root directory**, sometimes referred to as the "//" directory, or 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. Chapter 2 describes in detail how to maintain node root directories.

Upper-Level Directories

Upper-level directories are one level below the node entry directory in the DOMAIN naming tree structure. Upper-level directories contain the DOMAIN and DOMAIN/IX system software, system and programming libraries, and user's home directories. The system software installation procedures create the system-required upper level directories. In addition, you can establish upper-level directories on each node, such as home directories for different users.

You should protect the upper-level directories from unauthorized use by creating registries. (See Chapter 4.) You also must protect system software to maintain the node's ability to operate. Software protection is discussed in Chapter 5.

Disk Volumes and Volume Entry Directories

Disked nodes have one or more **physical volumes**, that is physical media. One disk drive is treated 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. However, most nodes will only have one logical volume per physical volume, which is the most efficient way to use disk space. You use the DOMAIN **invol** utility to create and maintain logical volumes on the physical media.

Logical Volumes on DOMAIN/IX Systems

DOMAIN/IX uses device descriptor files (in the */dev* directory) to define the logical volumes. The DOMAIN/IX installation procedure automatically creates three sets of disk device descriptor files, one each for a single Winchester, floppy, and storage module logical volume. You must use the **/etc/mkdisk**(8) command to create additional device descriptor 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 use **mkdisk** to create the following device descriptor files:

/dev/sm1a Block device descriptor for logical volume 1 (a) of the second storage module unit

/dev/sm1b Block device descriptor for logical volume 2 (b) of the second storage module unit

/dev/rsm1a Raw device descriptor for logical volume 1 (a) of the second storage module unit

/dev/rsm1b Raw device descriptor for logical volume 2 (b) of the second storage module unit

You use the device descriptor files in the **mount**(8) and **umount**(8) commands to mount and unmount the volumes. A volume must be mounted to be accessible. Any volume that you mount using the **/com/mtvol** command is also accessible to DOMAIN/IX, even if the volume does not have a descriptor file. However, the volume will not 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 does not include the boot volume, even though it is mounted.

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 cataloged in the next higher directory in the naming structure. The boot volume (node) entry directory is cataloged in the root directory, the highest level directory in the network naming structure. The **mount** command automatically catalogs a volume's entry directory in its next higher directory when you mount the volume. Similarly, the **umount** command uncatalogs the volume entry directory.

Note that, because the name of a directory is cataloged in 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, as 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 dismount it and mount a disk with financial analysis results as `/finances/q286`.

Figure 3–2 illustrates the node `apple` with two storage module drives. One physical storage module volume is a single logical volume. This volume is also the boot volume. The second storage module is partitioned into two logical volumes, with entry directories `/sm1/vol1` and `/sm1/vol2`.

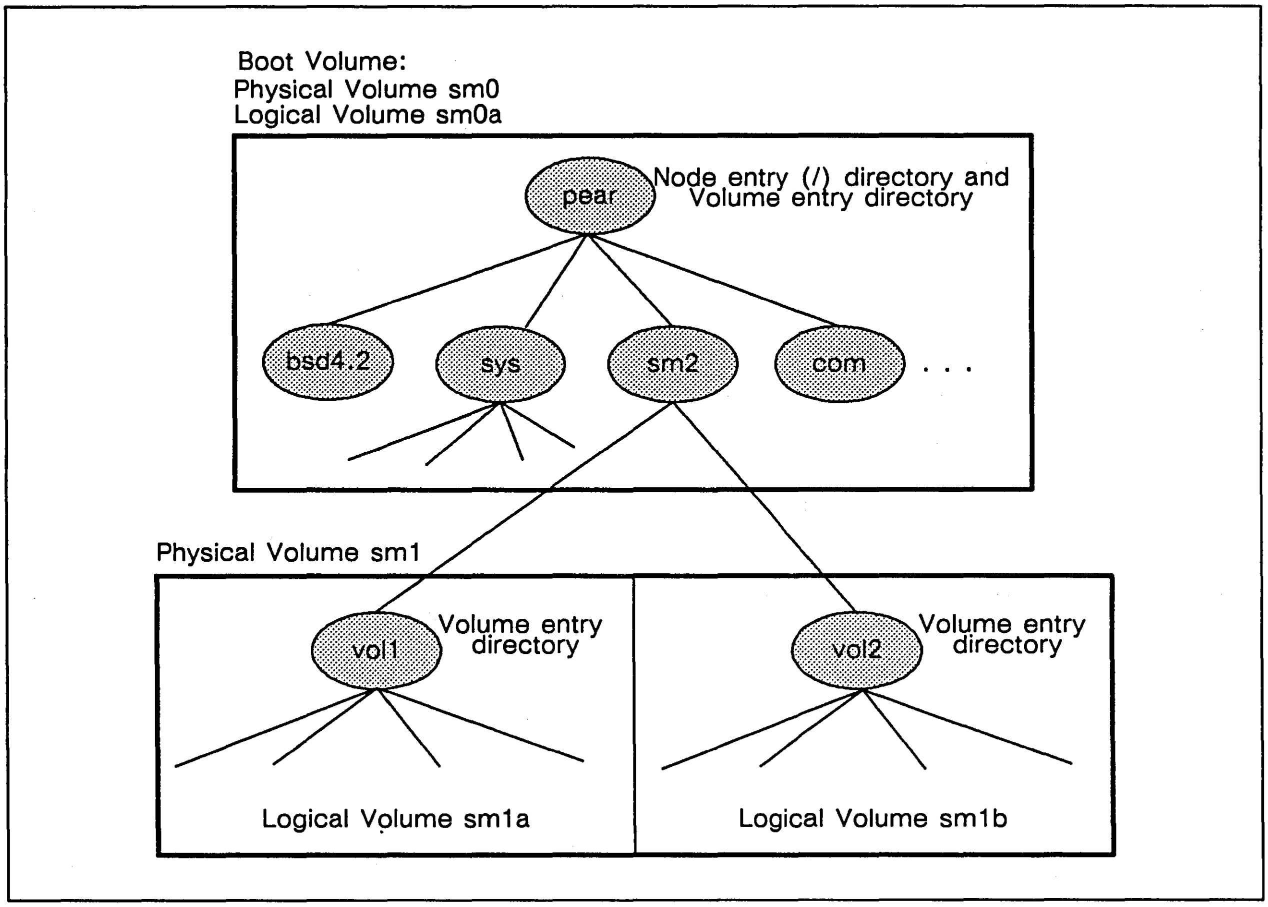


Figure 3–2. A node with Multiple Physical and Logical Volumes

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 you must:

1. Boot the node as a diskless node
2. Use the **mkdisk** command to make a device descriptor 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 **mount** command to mount the required logical volumes.

You must do step 2 because a diskless node's *'node_data* directory does not automatically have DOMAIN/IX device descriptors. For more information on the requirements for diskless nodes, see "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/mkdisk mount /dev/wn1a
% /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 workers at other nodes) can refer to the volume's files 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.

Directories and links

The *DOMAIN System User's Guide* describes directories and links in detail. However, there are several points that are particularly important to system administrators. These include:

- The meaning of 'node_data (tick-node data) and / (slash)
- The use of variant links in DOMAIN/IX
- The differences between symbolic links created with **ln**(1) and links created with **/com/crl**

'node_data and /

The meanings of the identifiers *'node_data* and / (at the beginning of a pathname) are context-dependent. As a result, when used incorrectly (particularly in links) they can result in circular references or references to the wrong files.

The slash character (/), when used at the start of a pathname, 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 / and *'node_data* conventions are powerful tools. The *'node_data* convention ensures that you use the node_data directory that belongs to the node where you are doing your work, and that node only, even if the node is diskless. Similarly, the / convention refers to your node's entry directory, independent of the current working directory.

Some examples of using / and *'node_data*

1. *'node_data* and diskless nodes.

Suppose there are three nodes: `bigdisk`, `nodisk`, and `diskless`. `bigdisk` is the partner node for `nodisk` and `diskless`. A program or person working at node `bigdisk` that reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data/startup.191`. A program or person working at node `diskless` that reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data.e467/startup.191`, where `e467` is `diskless`' node ID. A program or person working at node `nodisk` that reads the file *'node_data/startup.191* will read `//bigdisk/sys/node_data.632b/startup.191`, where 632b is nodisk's node ID.

2. **/com/crp**, **rlogin**(1), **rexec**(3x), **rsh**(1), and **rcmd**(3x)

If you use any of these utilities 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 `mynode` and use **rlogin** to log in remotely on the node yournode, the following command:

```
% ls /sys
```

displays the directory listing for `//yournode/sys`, the remote node's /sys directory.

3. 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 indeterminate or 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 `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, that is:

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

Variant Links

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 SYSTYPE environment variable. Therefore, if a process' SYSTYPE is *bsd4.2,* the process executes commands in the */bsd4.2/bin* and */bsd4.2/usr* directories.

This use of variant links allows different processes to use different versions of DOMAIN/IX simultaneously, and ensures that each process uses the correct versions of each command and call. For example, if you develop software that will be used under both *sys5* and *bsd4.2,* it enables you to correctly run a *bsd4.2* C shell and a *sys5* Bourne shell simultaneously.

Figure 3–3 illustrates that */usr* is a variant link to */bsd4.2/usr* or */sys5/usr.* "The DOMAIN/IX Environment" section discusses environment variables in greater detail.

Symbolic Links

The symbolic links created by the DOMAIN/IX **ln**(1) command and the **/com/crl** command are treated identically by most DOMAIN/IX and AEGIS commands. For example, you can delete either of these links with the **rm**(1) or the **/com/dll** command. However, the links are not identical objects. The **ln –s** command creates a new file which contains the linkage information, while the **/com/crl** command creates a link entry in the directory.

The **ls**(1) command does not show this difference between these links, but the **/com/ld** command, when used with the **–st** and **–tu** options (or the **–a** option), shows the link types. For example, if you execute the following commands:

```
% ln –s //joes_node/joe /joe
% /com/crl /helen //helens_node/helen
```

and then list the / directory using **ls** and **/com/ld** you will see:

```
% ls –l /
.
.
lrwxrwxrwx  1 hal            12 May  5 11:05 joe –> //joes_node/joe
lrwxrwxrwx  1 <none>         12 May  5 11:05 helen –> //helens_node/helen
.
% /com/ld / –tu –st
.
file  slink     joe
link            helen
.
.
```

Node Directory Structure

Figure 3–3 shows a typical node directory structure, including system–provided directories and user entry directories, starting at the node entry directory level. This figure only shows the general way the software is organized. It does not include all required system directories and does not show most links. The following section describes the DOMAIN and DOMAIN/IX system directories.

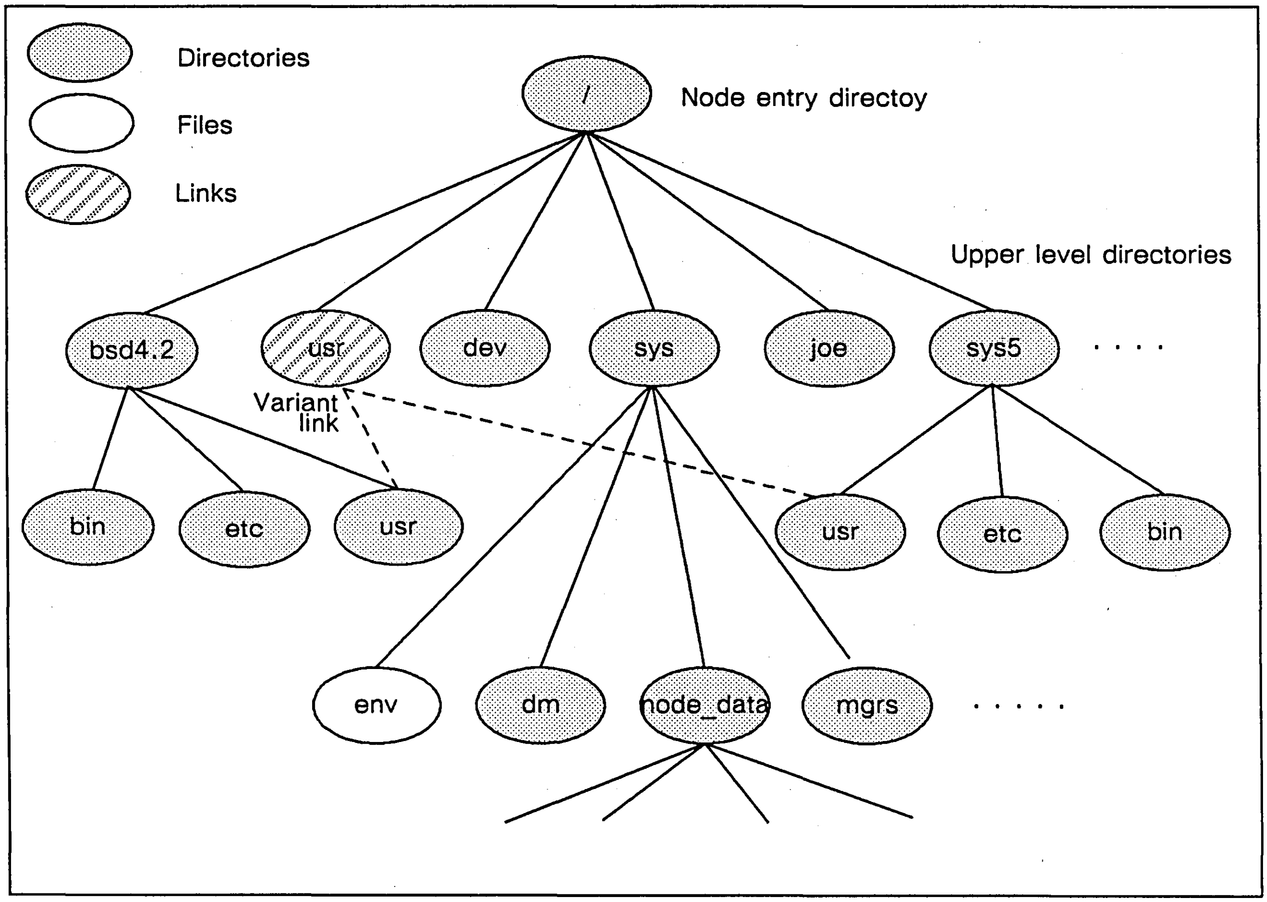


Figure 3–3. Disked Node Directory Structure

DOMAIN and DOMAIN/IX System Software Structure

DOMAIN and DOMAIN/IX software is organized in the node entry directory, which contains subdirectories and files. The following tables list the directories that are used to create and maintain the network environment. Table 3–1 shows the standard contents of the node entry directory (/) after you have installed DOMAIN/IX software. If you have optional DOMAIN software, the node entry directory may have additional entries. Table 3–2 shows the directories and files in the system software directory, */sys*, that are used frequently for network management.

Table 3–1. The Node Entry Directory (/)

Entry	Contents
bin	a link to SYSTYPE/bin
bscom	Boot shell commands
bsd4.2	*bsd4.2* SYSTYPE directory, contains *bin*, *etc*, and *usr*
com	AEGIS Shell commands
dev	a link to *'node_data/dev*, the node device file directory
doc	Release notes and update procedures
domain_examples	Example programs for documentation and tutorials
etc	A link to SYSTYPE/*etc*
install	Software installation scripts
lib	System libraries
preserve	Files saved during an installation procedure
registry	Created by the **/com/crrgy** (create_registry) procedures
sau[n]	Stand–alone utilities
sau1	for DN400/420/600
sau2	for DN300/320/330
sau3	for DSP80/80A/90
sau4	for DN460/660/DSP160
sau5	for DN550/560/570/580
sau8	for DOMAIN 3000 series
sys	System software
sys5	*sys5* SYSTYPE directory, contains *bin*, *etc*, and *usr*
sysboot	Node boot program
systest	On–Line test files
tmp	a link to *'node_data/tmp*, the node temporary file directory
usr	a link to SYSTYPE/*usr*

Table 3-2. The */sys* Directory

Entry	Content
alarm	Alarm server
apollo_logo	Apollo logo file; displayed on node start-up
boot	Initial process bootstrap program
cdict	Hashed word list for **/com/fserr**

color [type]_microcode[type]
color2_microcode
color3_microcode
color_microcode
color2_microcode.550
	Microcode for color workstations
	for DN 580
	for DN 570
	for DN600, DN660
	for DN550, DN560

ctboot	Cartridge tape bootstrap program
dict	Dictionary list for **/com/fserr**
dictdx	Dictionary index for **/com/fserr**
dm	Display Manager files
env	Bootshell environment bootstrap program
gpul_microcode.a , .b	Microcode for DN580 3D graphics accelerator
help	On-Line HELP files for AEGIS
ins	C, Pascal, and FORTRAN user insert files
mbx	**mbx_helper**, used for inter-process communication
mgrs	Type manager files
net	Network management files
node_data[.node_id]	Node-specific data files
ns	**ns_helper**, used for management of root directories
peb_microcode	Microcode for DN420 floating point processor
peb2_microcode	Microcode for DN320 floating point processor
sf	**sf_helper**, used for IPC store and forward
siologin	Serial I/O line servers
source	Source file for various programs
spm	Server Process Manager
subsys	Login protected subsystem
sysdev	SIO device descriptor files
traits	System trait files
types	Type definition file

Two subdirectories of the */sys* directory, the Display Manager directory */sys/dm* and the network management directory */sys/net*, are important when you creating and administering network services. Tables 3–3 and 3–4 show the contents of these directories.

Table 3–3. The */sys/dm* Display Manager Directory

Directory/File	Contents
bsd4.2_keys[n]	Standard key definitions for use with *bsd4.2*
bsd4.2_keys	for 880 keyboard
bsd4.2_keys2	for low–profile Model I keyboard
bsd4.2_keys3	for low–profile Model II keyboard
color_map	Color mapping information for color nodes
dm	Display Manager
fonts	Display Manager character fonts
input	Standard input file (a null file)
output	Standard output file (a null file)
sbpl	Tablet Server
startup_login[.type]	Node DM Start–up file
startup_templates	Node Start–up file templates
std_keys[n]	Standard key definitions
std_keys	for 880 keyboard
std_keys2	for low–profile Model I keyboard
std_keys3	for low–profile Model II keyboard
sys5_keys[n]	Standard key definitions for use with *sys5*
sys5_keys	for 880 keyboard
sys5_keys2	for low–profile Model I keyboard
sys5_keys3	for low–profile Model II keyboard
unix_keys[n]	Standard key definitions for use with either DOMAIN/IX
unix_keys	for 880 keyboard
unix_keys2	for low–profile Model I keyboard
unix_keys3	for low–profile Model II keyboard

Table 3–4. The */sys/net* Network Management Directory

File	Contents
diskless_list	List of the diskless nodes that can use this node as a partner
netboot	Diskless node bootstrap program
netmain_srvr	Network maintenance server
netman	Diskless node support server
sample_diskless_list	Example *diskless_list* file

The /etc Directory

Because of the distributed nature of the DOMAIN/IX environment, there are several considerations in the way you must administer system-wide and node-specific resources. This is particularly true with reference to the */etc* directory.

The */etc* directory contains administrative information. On most networks, there is only a single */etc* directory on an administrative node, and all other nodes link to this directory. (You can have multiple */etc* directories, but you must synchronize their contents, as described in the "Using Multiple Administrative and Service Nodes" section.) There are several entries in the */etc* directory, however, that must contain node_specific information. Therefore, these entries are links to the *'node_data* directory, the node_data directory of the node that is accessing the */etc* directory.

Figure 3–4 shows a listing of an */etc* directory. While it includes most files and directories that you would find in the */etc* directory, it does not necessarily contain entries required by optional or special purpose software. Similarly, some of these files are not required if you do not implement all DOMAIN/IX features. For information on individual files and directories, see the *DOMAIN/IX Programmer's Reference for bsd4.2* and *Managing TCP/IP–Based Communications Products,* as well as this manual.

```
% ls -l /bsd4.2/etc
otal 989
-rwxrwxrwx  1 root          2342 Feb  4 16:13 addroot
-rwxrwxrwx  1 root          4930 Feb  4 16:13 catman
-rwxrwxrwx  1 root          2548 Feb  4 16:13 chown
-rwxrwxrwx  1 root          7292 Feb  4 16:13 cron
-rwxrwxrwx  1 root         20956 Feb  4 16:13 crpasswd
-rwxrwxrwx  1 root          3310 Feb  4 16:13 crpty
-rwxrwxr-x  1 root          3872 Feb  4 16:13 cvtumap
-rwxrwxrwx  1 root            45 May 27 10:24 dmmsg
lrwxrwxrwx  1 <none>          17 May  5 07:09 etc.rc -> `node_data/etc.rc
-rwxrwxr-x  1 root          2148 Feb  4 16:13 fix_cache
-rwxrwxr-x  1 root          1174 Feb  4 16:13 flush_cache
-rwxrwxr-x  1 root         47350 Feb  4 16:13 ftpd
-rw-rw-rw-  1 alm              0 May  5 07:09 gateways
-rwxrwxrwx  1 root          3782 Feb  4 16:13 gettable
-rwxrwxr-x  1 root          4456 May  5 07:09 group
-rwsr-xr-x  2 root          4146 Feb  4 16:13 halt
-rw-rw-rw-  1 root          3173 May  5 07:09 hosts
-rwxrwxrwx  1 root           364 May  5 07:09 hosts.equiv
-rwxrwxrwx  1 root         25026 Feb  4 16:13 htable
-rwxrwxr-x  1 root         13070 Feb  4 16:13 inetd
lrwxrwxrwx  1 <none>          25 May  5 07:09 inetd.conf -> `node_data/etc.inetd.conf
-rwsr-s---  1 root         34282 Feb  4 16:13 lpc
-rwxrwxr-x  1 root          4548 Feb  4 16:13 mkdisk
-rwxrwxrwx  1 root          2987 May 28 15:57 mkptnr
-rwxrwxrwx  1 root            37 Mar 14 11:21 motd
-rwxrwxr-x  1 root          6890 Feb  4 16:13 mount
lrwxrwxrwx  1 <none>          19 May  5 07:09 mtab -> `node_data/etc.mtab
drwxrwxrwx  1 root          1024 Feb  4 16:13 net
-rw-rw-rw-  1 alm            390 May  5 07:09 networks
-rwxrwxr-x  1 root         11178 Feb  4 16:13 pac
-rwxrwxr-x  1 root        123877 May  5 07:09 passwd
-rwxrwxr-x  1 root         53576 May  5 07:09 passwd.map
-rwxrwxrwx  1 root           112 Feb  4 16:13 phones
-rwxrwxr-x  1 root           819 Mar  4 11:06 printcap
-rwxrwxrwx  1 root           289 May  5 07:09 protocols
lrwxrwxrwx  1 <none>          17 May  5 07:09 rc -> `node_data/etc.rc
lrwxrwxrwx  1 <none>          17 May  5 07:09 rc.local -> `node_data/etc.rc.local
-rwsr-xr-x  2 root          4146 Feb  4 16:13 reboot
-rwxrwxrwx  1 root          1181 Feb  4 16:13 remote
-rwxrwxrwx  1 root          2590 Feb  4 16:13 renice
-rwxrwxr-x  1 root         11762 Feb  4 16:13 rexecd
-rwxrwxr-x  1 root          6770 Feb  4 16:13 rlogind
-rwxrwxr-x  1 root          5542 Feb  4 16:13 route
-rwxrwxr-x  1 root         25494 Feb  4 16:13 routed
-rwxrwxr-x  1 root          8204 Mar 31 15:40 rshd
-rwsr-xr-x  1 root          2196 Feb  4 16:13 run_rc
-rwxrwxr-x  1 root         16768 Feb  4 16:13 rwhod
-rwxrwxrwx  1 root          1212 May  5 07:09 services
-rwxrwxrwx  1 root          2894 Feb  4 16:13 swapul
-rwxrwxrwx  1 root         10678 Feb  4 16:13 syslog
-rwxrwxrwx  1 root            27 Feb  4 16:13 syslog.conf
-rw-rw-rw-  1 root             3 Feb 12 17:32 syslog.pid
-rwxrwxr-x  1 root         14248 Feb  4 16:13 systype
-rwxrwxrwx  1 root             7 Feb  4 16:13 talkd
-rwxrwxr-x  1 root         10980 Feb  4 16:13 telnetd
drwxrwxrwx  1 root          1024 Feb  4 16:13 templates
-rwxrwxr-x  1 root         31157 Mar 27 16:38 termcap
-rwxrwxrwx  1 root         10922 Feb  4 16:13 tftpd
-rwxrwxr-x  1 root          6606 Feb  4 16:13 umount
-rwxrwxr-x  1 root          1490 Feb  4 16:13 update
-rwxrwxrwx  1 root          1030 Feb  4 16:13 update_slave
lrwxrwxrwx  1 <none>          19 May  5 07:09 utmp -> `node_data/etc.utmp
-rwxrwxrwx  1 root          1616 Feb  4 16:13 ver
-rwxrwxr-x  1 root          8042 Feb  4 16:13 writed
```

Figure 3-4. An /etc Directory Listing

The /sys/node_data[.node_id] Directory

Every node, whether it has a disk or is diskless, requires certain files that are used by that node only. These files include start-up and configuration files such as *inetd.conf* and *startup*[.type]; they also include per-node system files such as backing store for dynamic memory, inter-process communications mailboxes, and device files. The */sys/node_data*[.node_id] directory contains all of these per-node files.

The optional ".node_id" component in the file name enables a disked node to be a partner to one or more diskless nodes. In this case, the name of the diskless node's node_data directory includes the node's hexadecimal ID number. For example, if the disked node orange is the partner node for diskless nodes pear (node_id efd3) and cherry (node_id f2a4), orange would have the following three node-specific directories:

/sys/node_data	(for orange)
/sys/node_data.efd3	(for pear)
/sys/node_data.f2a4	(for cherry)

Table 3-5 shows a listing of a */sys/node_data*[.node_id] directory on a typical DOMAIN/IX node. While it includes most files and directories that you would find in the */sys/node_data* directory, it does not contain entries required by optional or special purpose software. Similarly, some of these files are not required if you do not implement all DOMAIN/IX features. Many of these files and directories are used only by system software. For information on files and directories that are used by specific application programs or utilities, see the documentation for that software. This documentation includes other chapters in this manual, the *DOMAIN/IX Programmer's Reference for bsd4.2*, and *Managing TCP/IP-Based Communications Products.*

NOTES: See "Diskless Node Administration", later in this chapter, for special considerations for the */sys/node_data*.node_id directory for diskless nodes.

Many files in the */sys/node_data*[.node_id] directory are accessed by links from other directories. See Table 3-6 for a list of DOMAIN/IX links.

Table 3-5. /sys/node_data[.node_id] Contents

File or Directory	Use or User
acl_cache	ACL to permission mapping
alarm_server.msg_mbx	Alarm server
boot_shell	Bootshell
boot_shell_template	Bootshell
cron	*sys5* **cron**(1M)
crontab	*bsd4.2* **cron**(8)
crp_mbx001	**/com/crp**
d3m_$error_log	D3M
d3m_$lm_shared_mem	D3M
d3m_$mbx	D3M
data$	System
dev	Device files, a link from /dev
dm_mbx	DM
etc.inetd.conf	*bsd4.2* **inetd**(8C)
etc.mnttab	sys5 **mount(1M)**
etc.mtab	*bsd4.2* **mtab**(5)
etc.rc	**rc**(8)
etc.rc.local	**rc**(8)
etc.utmp	**utmp**(5)

Table 3-5 (Cont.) /sys/node_data Contents

File or Directory	Use or User
global_data	System
global_rws	System
hint_file	System
ipc_data	System
mbx_$helper_lock	**/sys/mbx/mbx_helper**
networks	TCP/IP (configuration file)
null_hint_file	System
paste_buffers	DM (contains paste buffer files)
pdb	DM
preserve.bsd4.2	Installation procedures (saves old files here)
preserve.sys5	Installation procedures (saves old files here)
proc_dir	System (each directory entry is a process name)
proc_dump	System (process crash information file)
ptmp	DPSS/Mail, contains message sent
shell	Bootshell
shell.template	Bootshell
spm_mbx	SPM
stack	System
startup	DM (node startup file)
startup.1280color	DM (node startup file)
startup.19l	DM (node startup file)
startup.color	DM (node startup file)
startup.spm	DM (node startup file)
sys_error_log	System, error log
syslog	**syslog**(8)
sysmbx	System
talk_mbx	**talk**(1)
tcp_data	TCP/IP
thishost	TCP/IP (configuration file)
tmp	*/tmp* directory
usrtmp	*/usr/tmp* directory
vte_mbx	vt100 emulation
vte_mbx.ctl	vt100 emulation
writed_mailbox	**writed**(8)

System and Administrative Links

As we have indicated in previous sections, the DOMAIN/IX software structure includes a variety of links to specific files and directories. Many of these links are required because of the distributed nature of the DOMAIN system and the two *systypes* of DOMAIN/IX. Table 3-6 lists the system software links that are created when you install DOMAIN/IX. This table includes only DOMAIN links that are specific to DOMAIN/IX. It does not include any hard or soft links that are standard to *bsd4.2* or *sys5* software. It also does not include any links that you may install yourself.

Table 3–6. System and Administrative Links

Pathname	Links to
/bin	*$(systype)/bin*
/bsd4.2/usr/include/errno.h	*/bsd4.2/usr/include/sys/errno.h*
/bsd4.2/usr/include/signal.h	*/bsd4.2/usr/include/sys/signal.h*
/bsd4.2/usr/include/time.h	*/bsd4.2/usr/include/sys/time.h*
/bsd4.2/usr/lib/crontab	*‘node_data/crontab*
/bsd4.2/usr/preserve	*‘node_data/preserve.bsd4.2*
/bsd4.2/usr/spool/at	*‘node_data/usr.spool.at*
/bsd4.2/usr/ucb/mailq	*/bsd4.2/usr/lib/sendmail*
/bsd4.2/usr/ucb/newailiases	*/bsd4.2/usr/lib/sendmail*
/dev	*‘node_data/dev*
/etc	*$(systype)/etc*
/sys/node_data[.node_id]*/dev/tty*[0n]	*‘node_data/dev/sio*[n]
/sys/node_data[.node_id]*/dev/tty.spm*	*‘node_data/dev/sio1*
/systype/etc/etc.rc	*‘node_data/etc.rc*
/systype/etc/inetd.conf	*‘node_data/etc.inetd.conf*
/systype/etc/mtab	*‘node_data/etc.mtab*
/systype/etc/rc	*‘node_data/etc.rc*
/systype/etc/rc.local	*‘node_data/etc.rc.local*
/systype/etc/ttytype	*‘node_data/etc.ttytype*
/systype/etc/utmp	*‘node_data/etc.utmp*
/systype/usr/lib/cc	*/com/cc*
/systype/usr/lib/bind	*/com/bind*
/systype/usr/tmp	*‘node_data/usrtmp*
/sys5/usr/lib/cron	*‘node_data/cron*
/sys5/usr/preserve	*‘node_data/preserve.sys5*
/sys5/usr/spool/cron	*‘node_data/cron*
/tmp	*‘node_data/tmp*
/usr	*$(systype)/usr*

NOTES: *systype* indicates pathnames that are identical in the *sys5* and *bsd4.2* naming trees. See Tables 3–7 and 3–8 for links required on nodes that use *sys5* and *bsd4.2* and links required on replicated administrative nodes. See *Managing TCP/IP–Based Communications Products* for links required for TCP/IP communications.

The DOMAIN/IX Environment

This section describes elements of the DOMAIN/IX environment that are especially important to system administration.

Environment Variables

Environment variables are process–wide ASCII strings of the general form

NAME = value

that are generally used to store global state information. Environment variables are maintained by the kernel's process manager and are made available to both DOMAIN/IX and AEGIS programs. Child processes normally inherit the environment variables from the parent process. Whether a variable has meaning depends entirely upon the programs executing in the process. For example, the PATH variable is meaningful to the C and Bourne Shells, but has no meaning in the AEGIS shell. However, it could have meaning to a program executing in the AEGIS Shell.

The *DOMAIN/IX User's Guide* discusses environment variables in detail. The discussions of **environ**(7), **sh**(1) and **csh**(1) provide information on environment variables that are meaningful to DOMAIN/IX commands, the Bourne shell, and the C shell, respectively. The following sections discuss certain variables that are unique to DOMAIN/IX and are of particular interest in node and network administration. A following section discusses the way the DM inherits context, including environment variables.

SYSTYPE

The SYSTYPE variable determines the meaning of pathnames that include the */bin*, */usr*, or */etc* directory. These directories are variant links to system-dependent directories, such as */bsd4.2/bin*. By using the SYSTYPE variable and variant links, the DOMAIN/IX software ensures that you use the software that is appropriate for the version of DOMAIN/IX that you are using. For example, if the SYSTYPE environment variable is *bsd4.2*, then the pathname `/bin/ld` is interpreted as `/bsd4.2/bin/ld`, the *bsd4.2* version of the **ld**(1) command. If the SYSTYPE environment variable is *sys5*, then the pathname `/bin/ld` is refers to the file `/sys5/bin/ld`, the *sys5* version of **ld**. We discuss variant links in detail in the "Variant Links" section, above.

NAMECHARS

The NAMECHARS environment variable allows you to control the meanings of the following characters when they are used in pathnames:

Character	**Default Pathname Meaning**
\ (backslash)	Parent directory (when used anywhere in pathname)
~ (tilde)	Naming (home) directory. This character only takes on a special meaning when it is used as first character of pathname; it cannot be the first character of an object (directory or file) name, but can be included within the name.
‘ (backquote, or "tick")	Has a special meaning only in the directory name *‘node_data*, which refers to the node_data directory for the current node. This character cannot be the first character of any other object (directory or file) name, but you can include it within the name.

These characters have the default meaning in pathnames if:

- The process does not have a NAMECHARS environment variable; in this case all three characters have the default meanings.
- The NAMECHARS variable specifies the characters.

If you define a NAMECHARS environment variable and do not include one or more of these characters in the variable, then those characters will not have the default meaning and will be treated as normal pathname characters. You can then include them anywhere in file and directory names.

For example, if you include the following command in your *.cshrc* file:

```
setenv NAMECHARS '‘~'
```

The characters ‘ and ~ have the default meaning in a C shell, but the backslash character will have no special meaning.

You should always include the ' (tick) character in any NAMECHARS variable definition. If you define NAMECHARS without the tick character, the system software can not properly interpret references to the *'node_data* directory. This results in such problems as the failure to start server daemons at boot time (they are invoked from *'node_data/etc.rc*) and an inability to find */dev* (it is a link to *'node_data/dev*) and */tmp* (it is a link to *'node_data/tmp*).

NOTE: The NAMECHARS variable has no effect on a shell's interpretation of metacharacters. Therefore, you must always quote any pathname characters that are also metacharacters. For example, if you wish to specify the pathname `\testl` in the C shell you must enter either `'\testl'` or `\\testl`, independent of the value of the NAMECHARS variable.

Filename Mapping and Conversion

Versions of DOMAIN/IX software prior to SR9.0 used different name-mapping rules. Therefore, if you must access files whose names were generated by SR8 software (that have not already converted) you must convert the old names to follow the new rules. For example, if you restore archive tapes that were generated using SR8.1, you must then convert the files to the current name-mapping scheme.

Use the **cvtumap** command as follows to convert from SR8 to SR9 name mapping:

```
% /etc/cvtumap [-l] pathname(s)
```

where *pathname(s)* consists of one or more pathnames that you must convert from SR8 to SR9 name mapping. The **-l** option tells **cvtumap** to list the names of each directory and file when its name is remapped. **Cvtumap** will convert the mapping of the specified file or directory. It will also convert all names in the naming trees of the entered directories.

Remember, however, that the pathname arguments to this command must represent the (SR8-mapped) directory and file names as they are interpreted using SR9 mapping. Therefore, use one of the following techniques to ensure that you enter the pathnames correctly in the **cvtumap** command:

- Use the **ls** command to list the files and directories whose names you will use in the **cvtumap** command, and then enter the names exactly as they are displayed.
- Enter the name of the lowest-level SR9-mapped directory that contains all the files and directories to be remapped. In this case, **cvtumap** may display some error messages when it encounters names that already conform to SR9 mapping; however, it will do all mapping correctly and will not cause any errors in SR9-mapped names.
- Create an empty directory, restore or move all SR8-mapped files and directories to this directory, and then specify the directory in the **cvtumap** command. You can then move the converted files to their required locations.

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 and NAMECHARS values. It also includes the current working directory. This context inheritance has several importance effects, including:

- 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 *bsd4.2* C shell, move to a *sys5* 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 *sys5*, 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.

Using Both *sys5* and *bsd4.2* on a Node

DOMAIN/IX networks and nodes can use both *sys5* and *bsd4.2* versions of DOMAIN/IX simultaneously. Because DOMAIN/IX maintains parallel but separate *usr*, *bin*, and *etc* directories for the two systems, you can run most *sys5* and *bsd4.2* software simultaneously on a node. For example, you can simultaneously run a *bsd4.2* shell and a *sys5* shell, in separate windows. However, certain considerations and limitations are imposed when you use both *bsd4.2* and *sys5* software on a node. In particular:

- Certain files are not duplicated in the */bsd4.2* and */sys5* directory trees if you install *sys5* and *bsd4.2* on a node. Some files must not be duplicated on a node. Other files need not be duplicated because they are identical and would waste space if copies were located in both the */bsd4.2* and */sys5* directories. To prevent these duplications, the installation procedure creates links between entries in the */sys5* and */bsd4.2* directory trees; the actual files are put under the system type that you specify for **uucp** during the DOMAIN/IX administrative installation procedure. For example, if you specify *bsd4.2* **uucp** during the administrative installation, and use the **ls** command on an administrative node's */sys5/etc/passwd* file you would see:

  ```
  % ls -l //ad1/sys5/etc/passwd
  lrwxrwxrwx 1 <none> 22 May 28 14:08 //adl/sys5/etc/passwd -> //adl/bsd4.2/etc/passwd
  ```

  Table 3–7 lists the files and directories that are linked between types if you specify *bsd4.2* **uucp** during the system installation. For more information on these links see "Administering Networks with *bsd4.2* and *sys5*" and "Using Multiple Administrative and Service Nodes", below.

Table 3–7. Cross System–Type Links from */sys5* to */bsd4.2*

Linked Files and Directories	Comments
etc/passwd *etc/passwd.map* *etc/group*	All copies of these files must be identical. These files are linked between types only on nodes that have resident */etc* directories.
usr/bin/uucp */usr/bin/uudecode* *usr/bin/uuencode* *usr/bin/uulog* *usr/bin/uuname* *usr/bin/uusend* *usr/bin/snap* *usr/bin/uux* *usr/lib/uucp* *usr/spool/uucp* *usr/spool/uucppublic*	These files and directories are linked on all nodes. Only one version of **uucp** (*sys5* or *bsd4.2*) is allowed on a network. Therefore, the sys5 versions of **uucp** commands (such as */usr/pick*)are not supported if you specify *bsd4.2* **uucp** during system administrator install. However, all *bsd4.2* **uucp** commands are supported, independently of the shell's SYSTYPE, in this case.
usr/mail	Linked to */bsd4.2/usr/spool/mail*. Only one copy of this directory is allowed per network, all others must be links This directory is linked between types only on the node where the directory resides.

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 10. For information on **syslog**, see Chapter 11. For information on **tip**, see the *DOMAIN/IX Command Reference for bsd4.2.*

sendmail and syslog

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 **tcp_server** is not running on the **sendmail** node), then **sendmail** writes the message to *'node_data/syslog*.

tip

The following considerations apply to DOMAIN/IX **tip**:

- You can only use **tip** on a node that is physically connected to a modem.
- Each **tip** node must have a separate */usr/admin/aculog* file. At many installations, DOMAIN/IX 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*.
- 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.

Managing System Resources

You manage system resources by managing files and processes. When you administer a DOMAIN/IX network, you normally manage system-wide file resources, such as databases and libraries, and create each user's home directory. Similarly, you manage processes that provide network-wide services (server processes). If you use both *bsd4.2* and *sys5* on your network, you must manage the files and processes that are required by both versions of DOMAIN/IX, and you must manage processes and files that are distributed or replicated across the DOMAIN network.

Managing Network-Wide Resources

You must manage network-wide resources in order to:

- Distribute network resources such as databases
- Organize and protect user home directories
- Organize and protect libraries of application software.

Frequently, you do this 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 can be large or heavily used. You should position these resources strategically to maintain an even flow of network traffic and optimize disk space. The placement of servers and the selection of partners for diskless nodes are related issues in network management. If you are establishing a new network, you may wish to read the additional material in this book on network management before you locate upper-level directories that contain network resources.

You can create an upper-level directory for each user, which can become the user's home directory. A home directory is a default working and naming directory set by the operating system when the user logs in. Users' upper-level directories do not become their home directories until you create registries. Refer to the *DOMAIN System User's Guide* for a more detailed discussion of naming directories.

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

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**).

Providing System Services

Server processes are provide various system services to nodes on the network. Servers normally run regardless of log in and log out activity. These processes manage requests from **clients** that are programs or other processes. Clients request access to network resources like data, peripheral devices, or communication pathways outside the network. (There are also server processes or daemons that provide services to processes on an individual node; this section does not discuss these processes.)

Server processes often run on **Domain Server Processors (DSPs)**, server nodes that do not have displays and are dedicated to running processes that provide services to other nodes. You can configure network server processes on any kind of node whether or not there are DSPs in your network.

The number of times you implement a server process depends on the particular requirements of your site. 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.

You also must decide on where to place server nodes within the network topology. Decisions about the placement of server nodes are closely related to decisions about the number of times to implement a server. 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 locations, one in the small building and two in the larger building.

The location of server nodes 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.

Chapter 6 contains a general description of network servers and the server start-up attributes and options.The following sections in this chapter discuss special considerations for DOMAIN/IX servers and nodes that provide other network-wide DOMAIN/IX services. Later sections discuss the node start-up and log-in processes, and indicate methods for starting server processes at these times.

Administering Networks with *bsd4.2* and *sys5*

Because DOMAIN/IX maintains parallel but separate system directories, as a general rule, you can run both *sys5* and *bsd4.2* software and processes simultaneously on the network and on individual nodes. However, certain considerations are imposed by the incompatibilities of *sys5* and *bsd4.2* and by the resources they must share. "Using *sys5* and *bsd4.2* On a Node" discusses the considerations for running both systems on a single node.

In most cases, you can use both *bsd4.2* and *sys5* processes on a network. For example, you can use both the *bsd4.2* **lpd** line printer daemon, and the *sys5* **lp** line printer server on a single network. However, you can only use a single version of **uucp** on the network. If you install both *bsd4.2* and *sys5* on your network, the DOMAIN/IX installation procedure requests the version of **uucp** that you wish to install. It then automatically creates the links required so that you can use **uucp** independent of the system type, as shown in Table 3-7. As a result, if you use *bsd4.2* **uucp**, you can not use *sys5* **uucp** commands, but you can use *bsd4.2* **uucp** commands from a *sys5* shell.

Using Multiple Administrative and Service Nodes

Under DOMAIN/IX, certain services and administrative functions are provided a single node, or in some cases a limited number of nodes.These nodes include:

- System administrative nodes, which have copies of the */etc* directory.
- **uucp** server nodes, which run the **uucico.real** process
- Line printer spooler nodes, which run the *bsd4.2* **lpd** or *sys5* **lp** servers
- nodes that provide TCP/IP-based communications support
- The mail administrative node, which maintains the mail files

This section provides information on how to configure and manage these administrative and service functions on the network.

Configuring Multiple Administrative and Service Nodes

While a single node could provide all administrative and service functions, it is often useful to distribute functions among nodes and to have multiple nodes provide a particular network service, as follows:

- You can have multiple system administrative nodes. In a large network that includes several buildings, you could have an administrative node in each building that would service all user nodes in its building.
- You must have a **uucico.real** process at each node with communication lines used for **uucp**. Therefore you must have one such process per node with **uucp** dial lines.
- You can have multiple line printer spooler nodes, for example, one per DOMAIN network on a DOMAIN internet.
- TCP/IP support processes normally execute on each Internet gateway node.
- You must have a single mail administrative node.

Figure 3-5 shows a network with multiple administrative and service nodes.

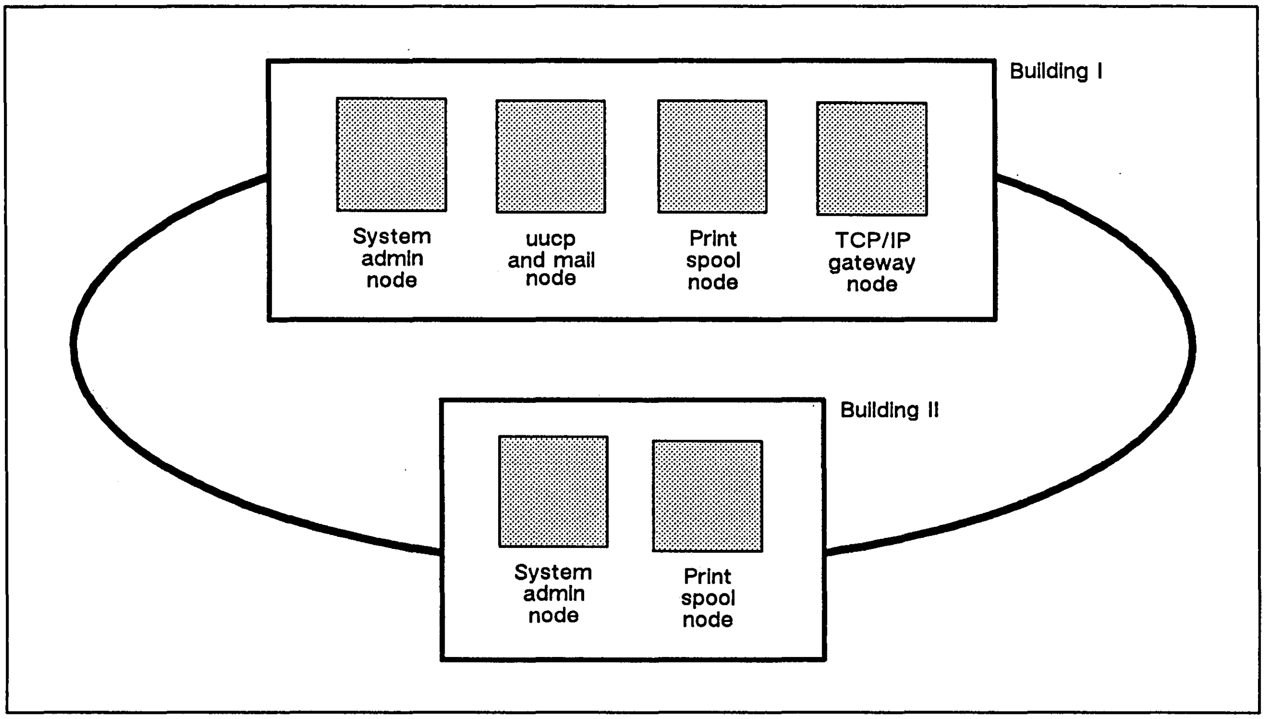


Figure 3–5 Network with Multiple Administrative and Service Nodes

Because of the distributed nature of the DOMAIN/IX system, you may run a DOMAIN/IX server process on more than one node. However, a DOMAIN/IX service may have a set of configuration or administration files associated with it that must be unique on the newtork. For example, you can have multiple nodes with **uucp** dial–up lines at a site, but they must all share the same **uucp** spooler and administrative files. Some administrative files (like */etc/passwd)* may be replicated to provide DOMAIN/IX system services across an internet, but all copies of the file contain the same information. DOMAIN/IX includes scripts which you can run periodically to update these files. (See **update_slave**(8).) Table 3–8 lists the files and directories used for various administrative and service purposes. It indicates the files' and directories' functions and specifies whether you can only have a single copy on the network or whether multiple copies are allowed.

Installing Software on Administrative and Service Nodes

The DOMAIN/IX system administrator's installation procedure creates copies of all the files listed in Table 3–8 on the target node. The user's installation procedure creates links to the source administrative node. If you want to provide only certain administrative and system services on a particular node, you must reconfigure the locations of the administrative and service files, and make sure that all links are correct. See Chapters 7 through 10 for details on configuration of line printers, TCP/IP, **uucp**, and **sendmail**.

Table 3–8. Administrative and Service File Links and Replication

Files and Directories	Comments
/etc/passwd */etc/passwd.map* */etc/group*	All copies of these files must be identical. These files may change frequently on a large network.
/etc/gateways */etc/hosts* */etc/hosts.equiv* */etc/networks* */etc/protocols* */etc/services*	All copies of these files must be identical. Used by TCP/IP. These files may change frequently on a large network
/usr/lib/aliases */usr/lib/aliases.dir* */usr/lib/aliases.pag* */usr/lib/sendmail.cf*	All copies of these files must be identical. Normally one per **sendmail**(8) daemon. All other nodes link to these files. These files may change frequently on a large network.
/usr/spool/mqueue	One per **sendmail**(8) daemon. Copies of this directory are not identical. All other nodes must link to this directory.
/usr/lib/uucp */usr/spool/uucp* */usr/spool/uucppublic*	One per **uucp** site; all nodes at the named site must link to the server node's files and directories. A site is normally the entire DOMAIN network or internet.
/etc/net/uname	All copies per uucp site must be identical.
/etc/printcap */usr/spool/lpd* */usr/spool/lpd.lock*	One per **lpd**(8) daemon; all nodes using this daemon must link to the daemon's directories.
/usr/spool/rwho	One per **rwhod**(8) daemon; normally one per DOMAIN, network but multiple daemons are allowed on a DOMAIN internet. Nodes using a daemon must link to its directory.
/usr/msgs	One directory per network; all nodes must link to a single node. Used by **msgs**(1).
/usr/spool/mail	One directory per network; all nodes must link to a single node. Used by **mail**(1) and **binmail**(1).

Synchronizing Replicated Administrative Files

As indicated in Table 3–8, many replicated administrative files must be identical. In most cases, these files do not change often; you can create all copies of these files once, and do not have to synchronize them unless you make changes. However, other files, such as */etc/passwd* can change frequently. In these cases, you must make sure that the files remain synchronized by making all changes on a single *master* administrative node. The master administrative node can then update the replicated files on all other administrative nodes by copying the changed files to the secondary, or **slave** administrative nodes on a regular basis.

Figure 3–6 shows the script, **update_slave**(8), that we provide to update slave administrative nodes. The DOMAIN/IX master administrative node's **cron** should run this script once a day for each slave node. For example, if a network has three administrative nodes, `//admin_master`, `//admin_slave1`, and `//admin_slave2`, then the `//admin_master` should have the following lines in its */usr/lib/crontab* file to update the slave administrative nodes:

```
/etc/crpasswd
0 6 * * * //admin_master/etc/update_slave //admin_master //admin_slave1
0 6 * * * //admin_master/etc/update_slave //admin_master //admin_slave2
```

The files are only synchronized immediately after they are updated. Therefore, any changes that you make during the day are not copied to the slave node until **cron** runs the */etc/update_slave* script (in the example, at 6:00 A.M). If you make any change that must be immediately available throughout the network (for example, if you register a new user who is located in another building) then you must manually run **update_slave** after you make the changes.

```
#!/bin/sh
# This script is run periodically to assure that each of the slave system
# administration nodes is up to date with the master system admin node.
# In general this script is run by cron on a regular basis with a line
# in the root crontab file with the following form:
#
# 0 6 * * * //master.node_id/etc/update_slave master.node_id slave.node_id
#
# In addition, this script may be run to update a slave sys admin node
# after a new user is added to the registry and the crpasswd file is run.
#
USAGE="USAGE: update_slave master.node_id slave.node_id"
if [ $# -ne 2 ]
then      echo $USAGE
          exit 1
fi
if [ ! -d //$1 || ! -d //$2 ]
then      echo $USAGE
          exit 2
fi
FROM=$1
TO=$2
PATH=/bin:/com
cd //$TO//etc
#first make sure passwd, passwd.map and group are all copied
rm -f passwd.last
mv passwd passwd.last
cpf //$FROM/etc/passwd
cpf //$FROM/etc/group -r -du
cpf //$FROM/etc/passwd.map -r -du

# For bsd4.2 systems copy the correct network files to the slave
netfiles="hosts hosts.equiv gateways networks protocols services"
for file in $netfiles
do
          if [ -f //$FROM/bsd4.2/etc/$file ]
          then      cp //$FROM/bsd4.2/etc/$file //$TO/bsd4.2/etc/$file
          fi
done
```

Figure 3–6. /etc/update_slave

NOTE: The version of **update_slave** that we provide only synchronizes required system files that are most likely to change with some frequency. You can edit this file to synchronize any other replicated files, including files listed in Table 3–8 or site–specific files such as */etc/motd* and */etc/profile.*

Start–Up Files

Several types of start–up command files exist. They are:

- Files executed by the operating system at node boot
- Files executed by the DM or SPM
- User files executed at login or shell creation.

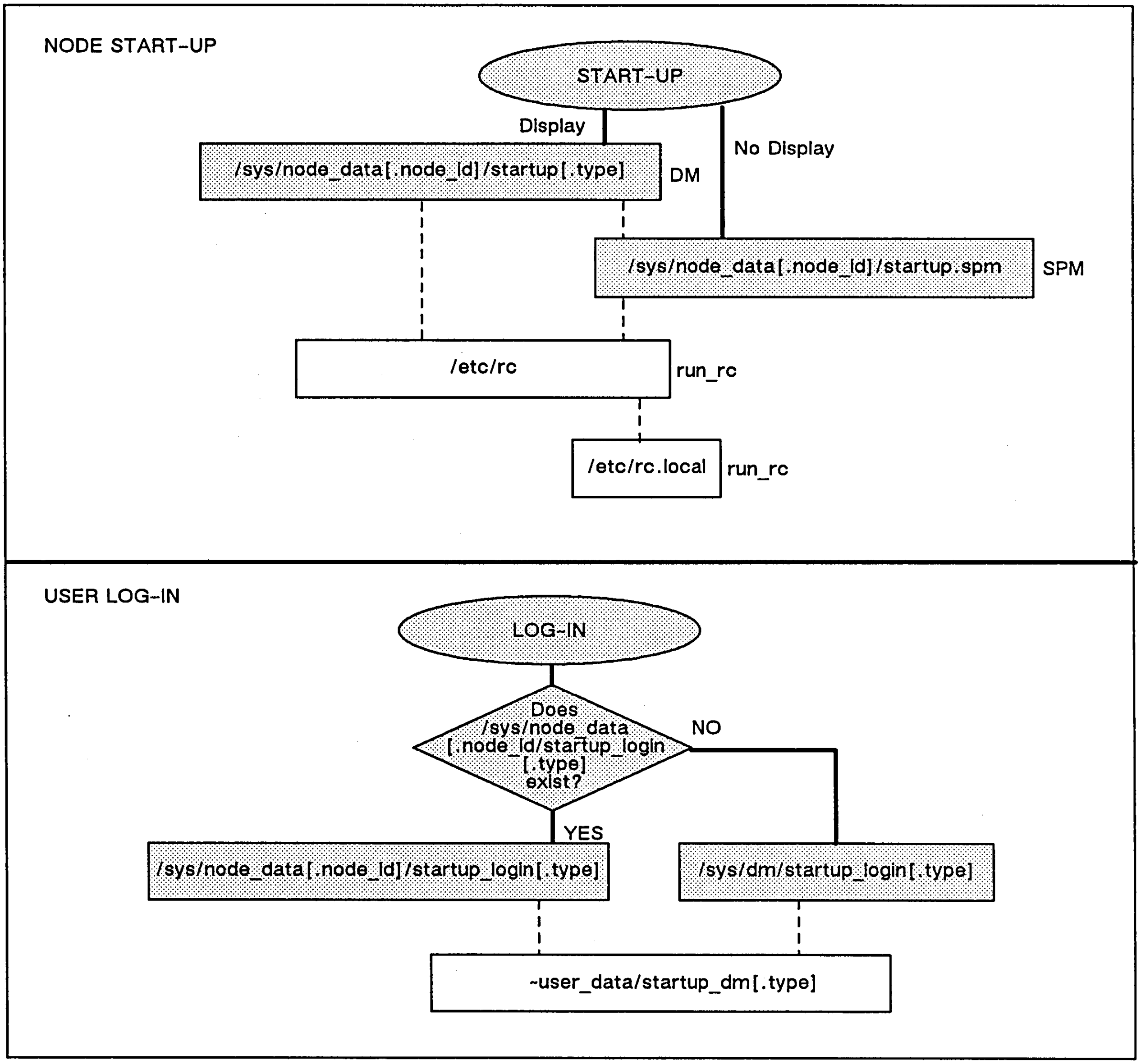
Table 3–9 lists the command files that can be used at these times. Some of these files run automatically; others must be specified by the automatic start–up files. The following sections describe the files that execute when the DM or SPM start running and when you log in to the DM. The shell documentation (in the *DOMAIN System User's Guide* and the *DOMAIN/IX User's Guide)* describes the shell start–up files and their functions in detail. Figure 3–7 illustrates the relations among the files and the start–up and log–in procedures.

Table 3–9. Standard Start–up Files

File	When Run	Comments
/sys/node_data[.node_id]*/startup*[.type]	boot	Runs automatically
/sys/node_data[.node_id]*/startup.spm*	SPM start–up	Runs automatically on DSPs
/etc/rc (a link to *'node_data/etc.rc)*	DM or SPM start–up	Must be specified by *startup*[.type] or *startup.spm*
/etc/rc.local (a link to *'node_data/etc.rc.local)*	DM or SPM start–up	Must be specified by */etc/rc*
/sys/node_data[.node_id]/ *startup_login*[.type]	User log–in to Display Manager	Runs automatically Always present, unless user–disabled
/sys/dm/startup_login[.type]	User log–in to Display Manager	Runs only if *startup_dm*[.type] is not present
~/user_data/startup_dm[.type]	User log–in to Display Manager	Must be specified in the node–wide DM start–up file
~/user_data/sh/startup	AEGIS Shell start–up	Runs automatically
~/.cshrc	C Shell start–up	Runs automatically
~/.login	C Shell started by */bin/start_csh*	Runs automatically Runs following *.cshrc*
~/.profile	Bourne Shell started *by /bin/start_sh*	Runs automatically

Notes: **.type** in a pathname represents a node display type identifier; A **.node_id** suffix to *node_data* represents the hex identifier of the diskless node for which the directory holds information.

This table does not include any configuration files, such as */etc/inetd.conf* or */etc/syslog.conf* that are required by processes that these files run. See the descriptions of the commands and processes for information on their required configuration files.


NODE START-UP
START-UP
Display
No Display
/sys/node_data[.node_id]/startup[.type]
DM
/sys/node_data[.node_id]/startup.spm
SPM
/etc/rc
run_rc
/etc/rc.local
run_rc
USER LOG-IN
LOG-IN
Does /sys/node_data [.node_id/startup_login [.type] exist?
NO
YES
/sys/node_data[.node_id]/startup_login[.type]
/sys/dm/startup_login[.type]
~user_data/startup_dm[.type]

Note: Solid lines (——) indicate automatic operations. Dashed lines (- - -) indicate paths where the preceding file must specify the following file (for example, /etc/rc must specify /etc/rc.local). Program names to the right of a file (e.g., run_rc or DM) indicate the program that executes the commands in the file.

Figure 3–7 Start–up and Log–in Files and Operations

Node Types and Start–up Files

Because of the differences among display formats, particularly because of differing 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.191* is the DM start–up file for nodes with an 800 x 1040 landscape display. Table 3–10 lists the display type suffixes and indicates the node models to which they apply.

Table 3-10. Start-up File Suffixes

File Suffix	*Node Types*
none	DN400, DN420 with portrait display
.1280bw	Monochrome DOMAIN 3000
.1280color	DN580
.191	DN300, DN320, DN330, DN460, DN550, DN560, DN 570, Color DOMAIN 3000, DN420 with landscape display
.color	DN600, DN660

Note that 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 at nodes with different display types.

New at SR9.5 - Logout Script Processing

The Display Manager at SR9.5 processes logout scripts. The logout script must exist in either the */sys/dm* or the *'node_data* directories, and the script must be named *startup_logout.?**, where *?** is the appropriate node type suffix. You cannot start up new processes with the DM **cp**, **cps**, or **cpo** commands from this script, although you may use the -w option to **cpo** or **cps**. Read the information on the -w option in the SR9.5 Release Notes very carefully before you attempt to use it, as nodes may hang if it is used incorrectly.

Template Files

The DOMAIN and DOMAIN/IX installation procedures automatically create several start-up template files. Table 3-11 lists these files and describes any special functions they have.

Table 3-11. Start-up Template Files

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	Not copied for diskless nodes; use **mkptnr**
/etc/templates/inetd.conf	Not copied for diskless nodes; use **mkptnr**

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; if you delete your start-up file you can copy and edit the template.
- The **netman** process copies them whenever it creates a */sys/node_data*.node_id file for a diskless node. For more details on this see "Administering Diskless Nodes", below.

The files in the */etc/templates* directory serve as master copies of the */sys/node_data*[.node_id]/etc.rc and */sys/node_data*[.node_id]/etc.inetd.conf files. However, **netman** does not automatically copy these template files when it creates a */sys/node_data*.node_id directory for a diskless node. As a result you must manually copy the files to a nawly created */sys/node_data*.node_id directory.

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:

#!/com/sh	For the AEGIS 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 DOMAIN/IX software contain most of the commands that you are likely need. However, most of these commands are commented out by starting the line with a pound (#) sign. You must delete the # from the start of the line to execute the command. For example, to run the **netman** process whenever the node boots, you would uncomment the following line in the */sys/node_data*[.node_id]*/startup*[.type] file by deleting the #:

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

NOTES: The DM, SPM and AEGIS Shell are currently case–insensitive. However, it is good practice to use lowercase characters when referring to system commands and files. For example, you should not rename the file */sys/node_data/startup.19l* to */sys/node_data/startup.19L.*

Any 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 methods described in Chapter 6.

DM/SPM Start–up Files

The DM/SPM start–up files are run whenever the Display Manager or Server Process Manager processes start executing. The DM normally starts running when you start or reboot a node that has a display. The SPM start–up file executes whenever you start the SPM process on any node.

The /sys/node_data[.node_id]/startup[.type] File

The */sys/node_data*[.node_id]*/startup*[.type] command file, where [.type] is any suffix except .spm, executes automatically when the DM starts running. You use it to determine the DM operating environment, to specify any server or background processes that must execute on the node, and to run start–up processes that, in turn, use additional start–up files.

You can use these scripts to start any server processes you want to run regardless of log in and log out activity. Typically, you start the processes used to provide network services from this script.

Figure 3–8 shows the start–up script we provide for nodes with 19–inch landscape (1024 x 800 pixel) displays. (We have deleted some blank lines to fit it on one page.) Corresponding scripts for other types of displays draw the locations of the DM windows in different places. Otherwise, the scripts are similar. These start–up scripts are well commented. You should be able to understand the purposes of most command lines from the comments, and later sections in the manual describe the individual server processes in detail. However, the following additional information can be helpful in modifying your files.

The cps -w Command

If you specify a –w option on a **cps** command to create a process in a start–up file, then the DM will wait until the new process terminates before processing any further commands in the file. The DM will also wait until the process completes before it processes any commands from the screen command line. Use **cps –w** whenever the process being created terminates after it completes its work and subsequent commands (or user actions after log–in) depend on the process completing.

For example, you should use this command in the */sys/node_data*[.node_id]*/startup*[.type] file to start the **run_rc** process. This ensures that **run_rc** completes its initialization procedures before you can log in. Figure 3–8 illustrates this use.

Do not use **cps –w** to start any processes, such as daemons and servers, that do not normally terminate. For example, if you use **cps** –w to start the **tcp_server**, the DM waits for the **tcp_server** to stop running and does not accept any further commands. As a result, the DM hangs and you must reset the node.

```
# STARTUP, /SYS/DM, default system startup command file for 191, 04/21/83

# Default is black characters on a white (or green) background.
INV -ON

(608,774)dr;(1023,799)cv /sys/dm/output
(556,774)dr;(608,799)cv /sys/dm/output;pb
(0,774)dr;(556,799)cv /sys/dm/input

# To enable the diskless node boot server, uncomment the
#              following CPS command.
# cps /sys/net/netman

# To startup default printer
# cps /com/prsvr  -n print_server

# To enable the summagraphic bitpad support, uncomment the
# following CPS command.
# cps /sys/dm/sbp1 /dev/sio2 L

# To startup mbx (IPC) helper
#cps /sys/mbx/mbx_helper

# To properly define the keys for the 880 keyboard,
# uncomment the following command.
#kbd

# To properly define the keys for the low-profile keyboard
# without the numeric keypad, uncomment the following command.
#kbd 2

# To properly define the keys for the low-profile keyboard
# with the numeric keypad, uncomment the following command.
#kbd 3
#DOMAIN/IX USERS:
#
# To use a DOMAIN/IX style login sequence, uncomment the following;
# env UNIXLOGIN 'true'

# If you DO NOT use UNIXLOGIN, but you still want to be put into
# your project list when you log in, uncomment the following line:
# env PROJLIST 'true'
#
# The file `node_data/etc.rc is a shell script that usually starts
# various DOMAIN/IX daemons (e.g., rlogind, cron) when the node is booted.
# If you want this script to be run whenever
# your node is booted, uncomment the following line
# cps -w /etc/run_rc
```

Figure 3-8. 'node_data/startup.191 Script

The /sys/node_data[.node_id]/startup.spm File

The */sys/node_data*[.node_id]*/startup.spm* command file executes whenever the SPM starts running. Therefore, this file can execute in two different circumstances:

- It is the only node start-up file that runs on a DSP.

- It runs in addition to the DM start-up file on nodes with displays that also run the SPM process. Because the SPM allows you to use the DOMAIN **crp** command to create remote processes, nodes such as DN660s that might be used as compute servers often run the SPM.

If the node does not have a display, the *startup.spm* file must do all required node initialization.

If the node has a display, the *startup.spm* file must not include the initialization commands that are already in the *startup*[.type] file. In fact, it is usually best to put all the node start-up commands in the *startup*[.type] file, and to make sure that there is no *startup.spm* file.

Figure 3–9 shows the first lines of start–up script that we provide in the */sys/node_data* directory when you install DOMAIN software. This script is similar to that in Figure 3–8, but it does not include definitions for creating Display Manager windows. You can edit this script to include any other process that you want to run, regardless of log in or log out activity.

```
# STARTUP.SPM, /SYS/SPM,
# default server process manager startup command file
#
# To make sure line 1 listens to XOFFs
#
cps /com/tctl -line 1 -insync
#
# To enable the diskless node boot server, uncomment the
# following CPS command.
# cps /sys/net/netman
#
# To startup default printer
#
# cps /com/sh -n print_server
#
# To enable the summagraphic bitpad support, uncomment the
# following CPS command.
#
# cps /sys/dm/sbp1 /dev/sio2 L
.
.
# D O M A I N / I X U S E R S :
.
.
.
```

Figure 3–9. ʹnode_data/startup.spm Script

The /etc/rc and /etc/rc.local Files

The */etc/rc* and */etc/rc.local* files contain commands to initialize the */tmp* directory, start any daemons, etc. The **/etc/run_rc** program runs the commands in these files. Therefore, you must specify **/etc/run_rc** in the */sys/node_data*[.node_id]*/startup*[.type] file. **Run_rc** exits when it has executed all the commands in the */etc/rc* file.

The */etc/rc* file contains commands that might differ from node to node. On all nodes, it is installed as a link to *ʹnode_data/etc.rc*. Each *etc.rc* file must be owned by **root** and have the setuid bit on. This file must contain the commands that clear the */tmp* directory and that start any daemons that require root access rights, such as **cron**. On system administrator nodes, */etc/rc* should contain any DOMAIN/IX start-up commands that are under the system administrator's direct control. It also must include the command to execute any */etc/rc.local* command file. Figure 3–10 shows an */etc/rc* file for a system administrative node.

```
#! /bin/sh
#
# /sys/node_data/etc.rc (DOMAIN/IX version of 01/27/86)
#
# This script is run by /etc/run_rc during node startup.
# Uncomment lines to provide the functionality
# you need.
# /etc/run_rc sets stdin to /dev/null
# stdout and stderr to /dev/console
#
echo "*****  Node startup at `/bin/date`  *****"
#
HOME=/; export HOME
#PATH=/bin:/usr/bin
# Uncomment the following line for bsd4.2
PATH=/bin:/usr/bin:/usr/ucb
#
# Uncomment the following line for bsd4.2
(cd /tmp; /usr/lib/ex3.7preserve -a)
#
# Uncomment the following line for sys5
# (cd /tmp; /usr/lib/ex3.9preserve -a)
#
# Uncomment the following to use cron or at (both bsd4.2 and sys5)
# $SYSTYPE is used to determine which cron system to use.
#
/etc/cron
#
# The following two lines should only be uncommented
# on a node that will run uucico (both bsd4.2 or sys5).
#
cd /usr/spool
rm -f uucp/LCK.*
cd /
#
# Uncomment the following lines if you want to run the bsd4.2 lpr
# line printer server.  This should only be run on the node
# that contains the local /usr/spool/lpd directory (bsd4.2 only).
#
# if [ -f /usr/lib/lpd ]; then
#          /usr/lib/lpd &
# fi
# Uncomment the following lines if you want to run the sys5 lp
# line printer server.  This should only be run on the node
# that contains the local /usr/spool/lp directory (sys5 only).
#
# cd /usr/spool/lp
# rm -f FIFO SCHEDLOCK
# cd /
# /usr/lib/lpsched
#
# Run syslog if needed for error logging (bsd4.2 only).
if [ -f /etc/syslog ]; then
           /etc/syslog &
 fi
#
```

Figure 3-10. An /etc/rc File

```
# Run routed, rwhod, and sendmail only on nodes
# connected directly to the ETHERNET (bsd4.2 only):
#
# if [ -f /etc/rwhod ]; then
#         /etc/rwhod &
# fi
# if [ -f /etc/routed ]; then
#         /etc/routed &
# fi
# if [ -f /usr/lib/sendmail ]; then
#         (cd /usr/spool/mqueue; rm -f lf*)
#         /usr/lib/sendmail -bd -q1h &
# fi
#
# Run writed if you want to use the write(1) program
# (both bsd4.2 and sys5).
#
# if [ -f /etc/writed ]; then
#         /etc/writed &
# fi
#
# Run inetd to manage other internet daemons
# (ftpd, rshd, rexecd, rlogind, and telnetd)
# (bsd4.2 only)
#
if [ -f /etc/inetd ]; then
          /etc/inetd &
fi
exit 0
```

Figure 3-10 (Cont.) An /etc/rc File

Uncomment the following line in the */sys/node_data*[.node_id]*/startup*[.type] file to execute the commands in */etc.rc:*

cps /etc/run_rc

The **/etc/rc.local** file enables individual users to modify the operation of **run_rc** without giving the user root privileges. It is normally not owned by root and does not have the setuid bit on. As with the */etc/rc* file, */etc/rc.local* is actually located in the file */sys/node_data*[.node_id]*/etc.rc.local* on each node. The administrative node's */etc/rc.local* entry is a link to *'node_data/etc.rc.local.*

Because **run_rc** executes with a user id of root, any command in the */etc/rc* file normally executes with root privileges. If security is not a concern on your network, you can have **run_rc** execute the commands in the */etc/rc.local* file with root user privileges by including the following command in the */etc/rc* file:

. /etc/rc.local

If security is a concern, you can use the **su** command to reset the user id to **user** (the unprivileged user) before the commands in the */etc/rc.local* file execute. In this case, you should invoke */etc/rc.local* by including the following command in the */etc/rc* file:

su user < /etc/rc.local

System Files Executed At Log In

After you log in to the DM at a node, the DM executes one of the following scripts. Note that SPM does not execute this script during remote log-ins.

1. */sys/node_data*[.node_id]*/startup_login*[.type] (where type is the display type)

2. */sys/dm/startup_login*[.type], only if there is no */sys/node_data*[.node_id]*/startup_login*[.type] file for the node

NOTES: The SPM does not execute either of these scripts on remote log-in.

See the discussion of **siologin** for information on startup files executed when you log in over an sio line.

The *startup_login*[.type] script should start the processes that need to be invoked every time someone logs in. Examples include processes that draw a window and create shell or run a user-specific log-in script. Processes specified in the *startup_login*[.type] file stop running when the user logs off.

If you want a specific node to have a specific *startup_login* file, put a *startup_login* file in the */sys/node_data*[.node_id] directory. For example, if you want diskless node e37d to have a special *startup_login* file, create a file named `/sys/node_data.e37d/startup_login.191`. In this case, the operating system runs the node-specific login file and does not run the file in */sys/dm* whenever someone logs onto node e37d.

If you do not put a *startup_login*[.type] file in */sys/node_data*[.node_id] directory, the operating system executes */sys/dm/startup_login*[.type], the default script supplied with your node. Therefore, it is unnecessary to create *startup_login* files for each node ID if a standardized file (for each display type) is sufficient. In any case, it is good practice to keep a */sys/dm/startup_login*[.type] file for each type, both as a template file and to support any diskless nodes that might use the disked node as a temporary partner.

Figure 3-11 shows the */sys/dm/startup_login.191* script that we provide. This script creates an AEGIS Shell process by default. You can use the default shell window, comment it out by adding a "#" sign, change it to draw the shell windows in a different location, or replace it with a command to start a Bourne or C Shell. Uncomment the last line in the file to run the per-user log-in command file.

```
# STARTUP_LOGIN (the per_login startup file in `node_data or /sys/dm)

# main shell whose shape is generally agreeable to users of this node
(0,500)dr;(799,955)cp /com/sh

# and the users private dm command file from his home directory's user_data
# sub-directory. Personal key_defs file is also kept in user_data by DM.
# cmdf user_data/startup_dm.191
```

Figure 3-11. /sys/dm/startup_login.191 Script

User Files Executed At Log-In

If you remove the comment character (#) from the last line in the */sys/dm/startup_login* script, the DM looks for, and executes, *~/user_data/startup_dm*[.type] (where ~ is the user's naming directory). If you remove the pound sign from the *startup_login* script but do not create a */user_data/startup_dm*[.type] file, an error message appears at log in. The SPM does not execute this script during remote log-ins. We do not supply a *startup_dm* file but you can create the script using DM commands. The *DOMAIN System User's Guide* provides more information about this script.

Message of the Day and Log-In

The message of the day (*motd*) does not appear when you log in to the DM (as it did in releases prior to SR9.5). Instead, the contents of the file */etc/dmmsg* is written to the DM message pad. The message of the day, */etc/motd,* is written to the shell output pad when you start a login shell by using **/bin/start_sh** or **/bin/start_csh**. However, the *motd* is not written if you run **start_sh** or **start_csh** with the **-n** option. Therefore, the following DM command will start a login-style Bourne shell without writing the message of the day:

```
cp /bsd4.2/bin/start_sh -n
```

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 sections describe the rules and techniques for managing diskless nodes and their partners. The last section in this chapter describes a procedure for configuring a diskless node's partner.

Diskless Node Operation

When a diskless node displays the log-in prompt, all the programs required for its operation are in place. While the *DOMAIN System User's Guide* 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 "request for volunteer" broadcasts. It checks its */sys/net/diskless_list* file for the requester's hexadecimal node ID. If the ID appears in the file, it answers the diskless node's request.
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_name*/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 Display Manager executes the commands in the //partner*/sys/node_data*.diskless_node_id/*startup*[.type] file.

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

Establishing Diskless Nodes and Partners

A diskless node's partner node provides the system software and disk services for the diskless node. It 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 user nodes with display monitors, or DSPs, the server nodes.

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 */sau2* directory and a */sau5* directory. Similarly, the partner's */sys* directory must have any microcode files required by the diskless node. Tables 3-1 through 3-3 list many of the node-type dependent system software files.

Each diskless node has its own */sys/node_data*.diskless_node_id directory on the partner node, and *'node_data* on each diskless node resolves to */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

You control the assignment of diskless nodes to partners through the */sys/net/diskless_list* file, such as the one shown in Figure 3–12. A disked node can be a partner to several diskless nodes. The disked node will volunteer to be a partner for any diskless node whose node ID is in the disked node's *diskless_list*. 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.

```
# 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 3–12. A /sys/net/diskless_list

NOTES: If you put a diskless node 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 therefore its contents, could change whenever the node reboots. Therefore, you must configure the diskless node correctly on each possible partner node. (See the "/sys/node_data.node_id Directory on New Partners" section below for more details.)

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 the "Requesting a Specific Partner" section, below.

/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, with the minimal set of files required by the diskless node. **Netman** also copies the *startup*[.type] file from the partner's */sys/dm/startup_templates* or */sys/spm/startup_templates* directory. However, **netman** does not create the files required by DOMAIN/IX. As a result the diskless node will execute the template start-up script and will only run AEGIS software. Once netman creates the */sys/node_data*.node_id directory, you can use the following steps to ensure that the diskless node will start DOMAIN/IX correctly.

1. Run the **/etc/mkptnr** command script. This script creates null *etc.mnttab*, *etc.mtab* and *etc.utmp* files that are required by DOMAIN/IX, and copies template files to create the following start-up and configuration files.

   - */sys/node_data*.node_id/*etc.rc*
   - */sys/node_data*.node_id/*etc.inetd.conf*
   - */sys/node_data*.node_id/*thishost*
   - */sys/node_data*.node_id/*networks*

2. Edit */sys/node_data*.node_id/*startup*[.type] and the start-up and configuration files that were created in step 1 to meet the diskless node's needs.

3. Create any device descriptor files that you might need in the */sys/node_data*.node_id/*dev* directory. If you require pseudo-tty's, use the **crpty**(8) command. Use the **mkdisk**(8) command to create any required disk device descriptor files; you need these files if you are booting a node that has a disk as a diskless node and you wish to access the contents of the node's disk(s).

4. Create the */sys/node_data*.node_id/*etc.rc.local* file, if you use it.

The following section describes a procedure to provide a partner node that includes these steps.

Procedure: 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. Note that if a service representative installs a diskless node, he or she creates a partner for the node; however, you might want to use a different partner.

PROCEDURE 3-1: Providing a Permanent Partner for a Diskless Node

1. Determine the diskless node's ID number. 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:

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

2. Log in to the partner node. If the partner does not have a display, you can create a remote C shell on the partner by using the following command:

   ```
   % /com/crp /bsd4.2/bin/start_csh -on //partner -me
   ```

3. Add the diskless node's node ID to the partner node's */sys/net/diskless_list* file.

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

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

   a. Remove the comment character from the following line in the partner's */sys/node_data*[.node_id]*/startup*[.type] file:

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

      netman will now start automatically whenever you reboot the partner.

   b. Start the partner node's **netman** server. (This way you do not have to reboot the partner.) If you are at the partner, enter the following command in its DM input window:

      ```
      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 step 2.)

      ```
      % /com/crp -on node -cps /sys/net/netman -n netman
      ```

6. You can limit the size of the partner's memory pool that is available to the diskless node for paging requests. Enter:

   ```
   % /com/netsvc -p [pool_size]
   ```

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

   (The diskless node copies information (operating system, global data, etc.) in 1024-byte pages from its partner as needed. The default allows the partner's entire memory to be available for page requests from remote nodes. The minimum number of pages you can specify is 101.)

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

   - If you do not use **ns_helper** on your network, enter the following command. After you finish this procedure, catalog the node on the network as described in Chapter 2.

     ```
     % /com/ctnode node_name node_id
     ```

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

     ```
     % /com/ctnode node_name node_id -root
     ```

Procedure 3–1 (continued)

8. Create the diskless node's node start–up and configuration files. While there are many different ways you can do this, the following steps will work in all cases.

   a. 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.

   b. Run the **mkptnr** script as follows:

   ```
   % /etc/mkptnr -d node_id
   ```

   Note: If you are changing the partner of an existing diskless node, you can skip steps c and d. Instead, copy these configuration files from the old partner to the new partner.

   c. Copy the*/sys/node_data*.node_id*/startup*[.type] file (where type indicates the type of display on the diskless node, not the partner) from the partner. You can copy the file from the partner's */sys/node_data* directory, or from the */sys/dm/startup_templates* directory (for diskless nodes with displays) or */sys/spm/startup_templates* directory (for diskless DSPs) .

   d. Edit the following files to meet the diskless node's needs. For example, the *thishost* and *networks* files must include the diskless node's host name and Internet address, respectively.

   /sys/node_data.node_id*/startup*[.type]
   /sys/node_data.node_id*/etc.rc*
   /sys/node_data.node_id*/etc.rc.local*
   /sys/node_data.node_id*/etc.inetd.conf*
   /sys/node_data.node_id*/thishost*
   /sys/node_data.node_id*/networks*

9. Log off the partner node.

10. You can now start or restart the diskless node. If you are changing an existing diskless node's partner, do the following:

    a. Enter the following DM command to log off and shut down the operating system.

    ```
    Command: shut
    ```

    b. The Bootstrap PROM prompt (>) appears on the screen, enter the following reset command:

    ```
    > RE
    ```

    c. Press <RETURN> again.

    d. The PROM identifier appears, followed by the PROM Prompt. Restart the operating system by entering the following command.

    ```
    > EX AEGIS<RETURN>
    ```

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

END OF PROCEDURE 3–1

Managing Diskless Nodes and Partners

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

Diskless Node Management Commands

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. See *Administering Your DOMAIN System* for a detailed description of **netmain_srvr** and **netmain**.

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 reset 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 is good practice for the administrator of a partner node to notify diskless node users of intended shutdowns. Use the **/com/send_alarm** command with the **-di -mydi**, or **-din** option to notify users of shutdowns. For example, if you are shutting your node, `disked`, down, the following command will warn all diskless nodes that use your nodes as a partner:

```
/com/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 another disked node that runs **netman** as its partner, even if that node does not have the diskless node on its partner list. You can also use this procedure to boot a node that has a disk as a diskless node, for example if the node's system software is corrupted. You should not use this procedure in place of the *diskless_list*. 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 **shut** command to enter the Mnemonic Debugger (MD).

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

   ```
   > DI -N node_id<RETURN>
   ```

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

3. Enter the following command to restart the node.

   ```
   > EX AEGIS<RETURN>
   ```

NOTE: This procedure does not configure the partner's */sys/node_data*.node_id directory for proper execution of DOMAIN/IX.

Chapter 4

Creating and Maintaining User Accounts

The registry is a distributed database that identifies legitimate users of the network. The registry allows you to:

- Protect the system from unauthorized use by persons without valid accounts
- Provide users with home directories, and automatically start processes for them when they log in.

A network registry is not required, but networks without registries are open, and you cannot control user access in an open network. Because of this, we assume that most system administrators will choose to create a registry, in order to maintain a secure network. If your site has an internet, there are considerations for managing registries that are beyond the scope of this book. Refer to *Managing DOMAIN Internets* for information on registries in multiple ring environments.

In addition to providing security for your network through registries, you can protect individual nodes in the network with procedures described here. This chapter describes:

- How the registry controls the user's access to the network
- The parts of the registry database
- How to create and maintain a network registry at your site.
- How to generate and administer the DOMAIN/IX */etc/group*, */etc/passwd*, and */etc/passwd.map* files, and how to use the **crpasswd** command.

System Operation with the Registry at Log In

When a user logs in, the operating system looks at the files that form the registry database. These files are called the **person**, **project**, and **organization** files (**ppo**), and the **account** file (**acct**). You can think of the **ppo** files as a "master menu" that contains all the responses to a log in prompt that are possible at your site. For example, the **person** file contains all the log in names at your site, the **project** file can contain the names of projects or products, for example, "System_backup" or "Widget_design". The **organization** file can contain the names of groups, e.g., "Laboratory", "Marketing", or "Accounting." An optional **full_names** file, to list the full names of users associated with their log in account names, is also part of the registry database.

The **account** file contains those selections from the **ppo** menu that make up a user's log in account(s). For example, user John Jones might have several accounts under the same or different log in names. Each of his accounts can specify a different home directory and can have a different start-up script. In another case, any number of users (log in accounts) can share the same project name. Those users might want their working directory set to the directory of their project at log in. You arrange these options for users through the registry **account** file.

The **account** file also contains the user's password. You can put passwords in user accounts or users can create their passwords and change them from session to session. If the user changes a password, the operating system updates the **account** file.

If the account given to the log in prompt is not in the **account** file, the system refuses to allow the user to log in. However, the **account** file contains the default log in name "user.none.none". Any person can log in to the network as "user.none.none", or simply "user". With the user identification provided by the registry, however, you can control access to files and directories on the network.

Since it performs these important "gatekeeping" functions, the registry database or **site directory** (*/registry/rgy_site*) must be available at all times. It is not practical to rely on a single node to be in the network at all times. Therefore, the registry creation procedure distributes site directories to the nodes that you specify in the **master registry** file.

The master registry file (*/registry/rgy_master*) is a master list of the pathnames of site directories. The shell command **/com/lrgy** (list_registry) displays the master registry file once you have created the registry. Figure 4-1 shows an example of the master registry file.

```
 % lrgy<RETURN>
Registry:
//rose/registry/rgy_master Sites of registration data files:

           //rose/registry/rgy_site1
           //tulip/registry/rgy_site2
           //lilac/registry/rgy_site3
```

Figure 4-1. Example of a Master Registry File

To locate the registry sites, every node in the network has its own copy of the master registry file. The node's master registry file copy is called */registry/registry*.

When a user tries to log in, the system reads the node's registry file copy to obtain the pathname of a registry site directory. The system looks for the first available registry site directory on its list. Next, it checks the site directory's **account** file to verify the user's identity. When a user enters a valid log in name and associated password, the system:

- Sets the working directory to the home directory
- Sets the naming directory to the home directory
- Executes the user's start-up files if they exist

- Updates the node's local registry.

Figure 4–2 shows the actions of the operating system as a user logs in.

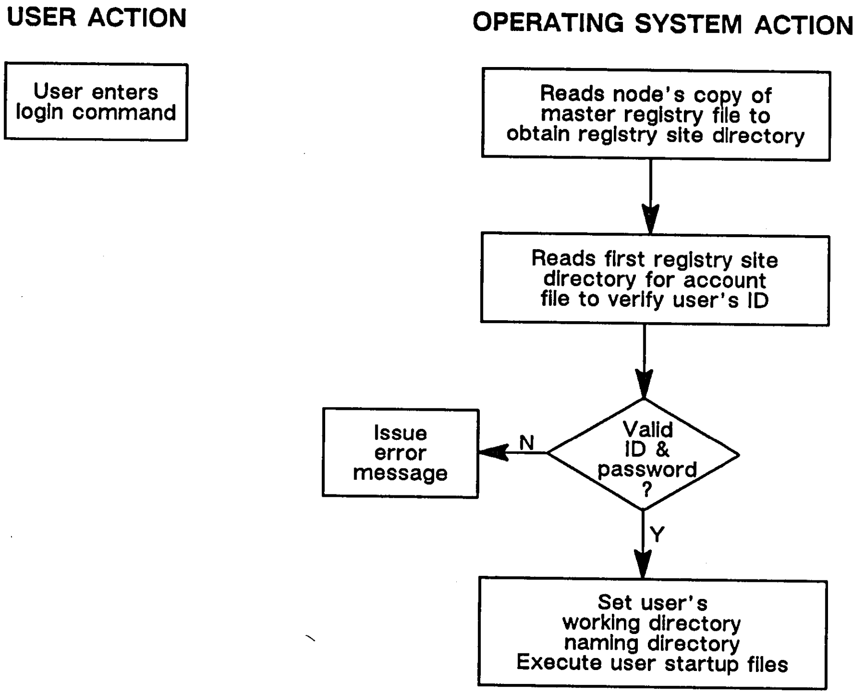

USER ACTION
OPERATING SYSTEM ACTION
User enters login command
Reads node's copy of master registry file to obtain registry site directory
Reads first registry site directory for account file to verify user's ID
Valid ID & password ?
N
Issue error message
Y
Set user's working directory naming directory Execute user startup files

Figure 4–2. Operating System Action During Normal Log In

The Local Registry

If a user logs in when the network registry is unavailable, i.e., the system tries to find the registry sites listed in */registry/registry* but cannot reach any of them, it uses the **local registry** as a last resort.

The local registry is a system maintained database that describes the node's last set of users and records the date and time when they last used the system. You create the local registry whenever you bring a node into the network. Each local registry resides in the node's */registry* directory.

The local registry has one registry site directory (*/registry/local_site*)and one master registry (*/registry/local_registry*) file. A local registry resides on every node in the network. The **account** file in */registry/local_site* stores the log in accounts of the node's last users.

The system maintains the local registry database. Every time someone logs in using the network registry, (i.e., "normal operation"), the system updates */registry/local_site*, on that node, with current information from the network registry database.

When the system uses the local registry, it reads */registry/local_registry* to get the pathname */registry/local_site*. It then checks the **account** file in the */registry/local_site* directory. If the user has logged in to the node recently with the account given to the log in prompt, the system finds the information in the **account** file and logs the user in. If the **account** file has no information on this user's account, the system refuses to log the user in.

If both the system registry and the local registry are unavailable, the system will automatically log the user in as "user.none.none," the system–wide default identity. This mechanism guarantees that you can log in at all times.

Figure 4–3 illustrates how the operating system uses the network and local registries to grant access to the system.

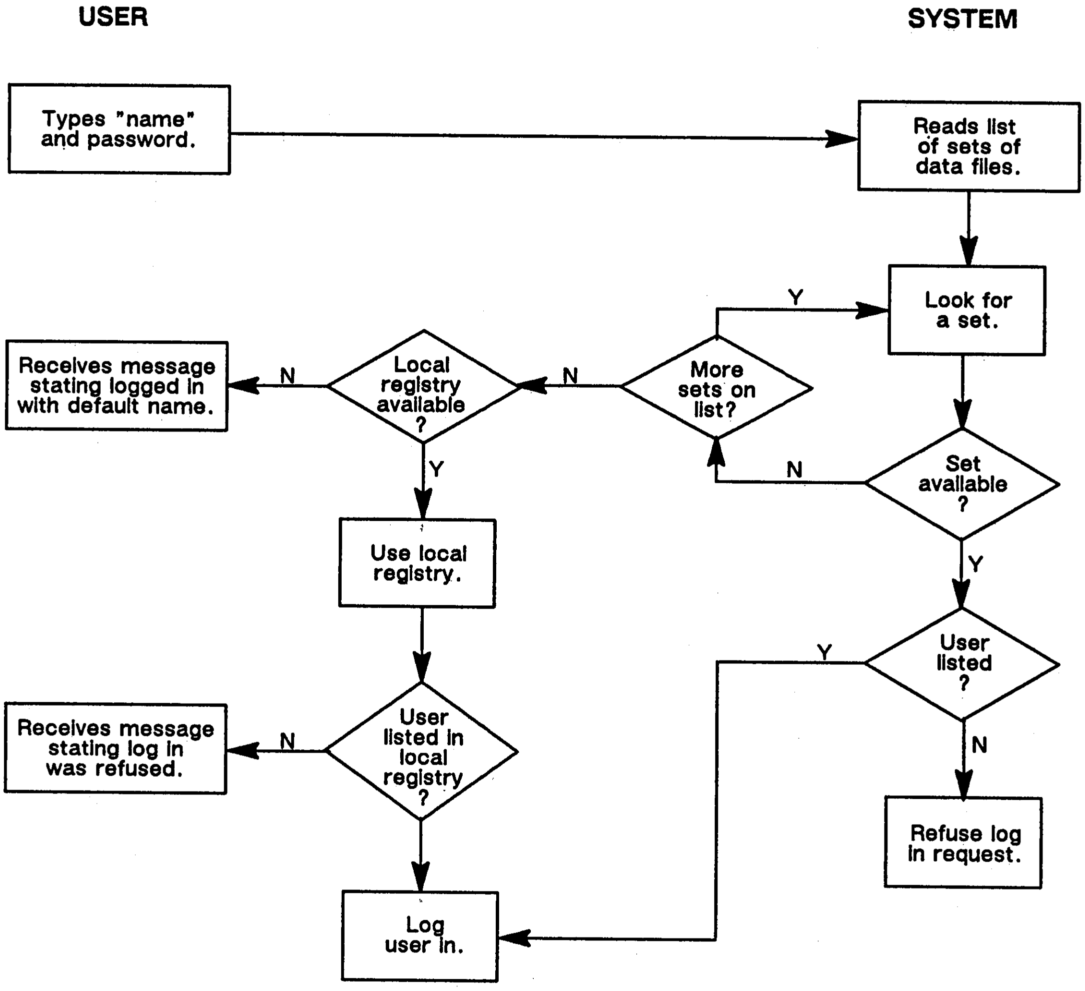

USER
SYSTEM
Types "name" and password.
Reads list of sets of data files.
Look for a set.
Y
More sets on list?
N
Set available ?
N
Y
Receives message stating logged in with default name.
N
Local registry available ?
Y
Use local registry.
User listed ?
Y
N
Refuse log in request.
Receives message stating log in was refused.
N
User listed in local registry ?
Log user in.

Figure 4-3. Using The Local Registry To Grant Access

Considerations for Implementing Your Network Registry

As system administrator, you must make several decisions about the network and local registries, and you must ensure that access to your registry is properly controlled. Before you create the registry, decide on:

- The number of registry site directories
- Registry site directory locations
- Registry site names
- Registry master file location
- Number of user accounts stored in local registries
- Time period of valid accounts in local registries.

Most installations will find two or three site directories sufficient to protect access to registries in case of node failure. If you have a multi–loop network, place the registry site directories on different loops to maintain registry service as loops are switched out of the network. Users on loops with registry site directories always have access to a site. Users on other loops will have access to at least one other site. The local registry can maintain service for most users when loops without site directories are switched out of the network.

Select stable nodes that always maintain their network services (**netsvc –a**) as registry site nodes. To minimize the possibility that a node will not maintain its network services, ensure that users of these nodes do not use the **netsvc** command with **–n** or **–l** options. These options prevent the node from responding to network requests that originate at other nodes. You can use ACLs to prevent any user from executing the **netsvc** command at the local node. A later section of this chapter describes how to set ACLs on the local copy of **netsvc**.

When you edit the registry site data files, you can do so successfully only when all site directories are available. Therefore, a loop or node that is often switched out of the network is not a good location for a registry site directory.

If you change the location of a site directory, update */registry/rgy_master*, the master list of pathnames. Also update each */registry/registry* in the network (the node copy of the registry master file). If a node (or loop containing the node) is off the network at the time of the update, its copy of */registry/registry* won't get updated. The problem this situation can create may not become apparent until long after you've done the update.

As described above, during a log in, the system looks in the list of pathnames for the first site directory, then looks for the next pathname if it doesn't find the first directory. Assume that some node's */registry/registry* file contains information that is only partially current. During network difficulties, the operating system might not reach any of the current site directories. Moreover, because *//node_name/registry/registry* contains outdated information, the operating system has fewer nodes to search for current site directories. A user might have difficulty logging in.

If the same node fails to receive new information over the course of several updates, its */registry/registry* file can become so outdated that some user won't be able to log in. To prevent this, follow proper registry maintenance procedures which include updating */registry/rgy_master* and every */registry/registry* each time you change site directory locations. Of course, you should try to prevent these occurrences by selecting good locations for registry sites and keeping the registries at those locations.

Although you cannot add names to the local registry, you can use the shell command /**com/crrgy –loc [n] –days [n]** to specify the number of accounts to be stored in the local registry and the time period that entries in the local registry will remain valid. You can use the shell command **lrgy** to display the local registry specifications on any node. Figure 4–4 shows the output of this command.

```
% lrgy -loc<RETURN>
Registry: //rose/registry/local_registry
Sites of registration data files:
        //rose/registry/local_site
Registry is LOCAL. It has 25 slots for login;
the expiration period is 21 days.
```

Figure 4-4. The Local Registry

You can also replace (delete and re-create) the local registry. Note that you lose the history of the node's last users if you replace the local registry. A new history will begin building immediately. Use **/com/edacct -l -loc** to examine the local registry on any node. Figure 4-5 shows the output of this command.

```
% edacct -l -loc<RETURN>
martin   none      eng     52D1 85/04/22.08:48exp:85/05/14 //jay/martin
marks    dev       lab     52D1 85/04/11.19:10exp:85/05/03 //robin/marks
user     none      none    52D1 85/04/19.10:39exp:85/05/11 /
brown    sys_admin eng     52D1 85/04/01.17:36exp:85/04/23 //duck/brown
```

Figure 4-5. The Local Registry account File

Because the system only performs updates to */registry/local_site* on the current node, the following scenario is possible:

1. Assume node A does not contain a network registry site directory. A user logs in at node A with password "X."
2. The same user logs in at node B and changes the password to "Z."
3. The user logs in at node A when the network registry is unavailable and the system must use the local registry. The node A local registry requires the password "X," not "Z," because the system does not update node A's local registry when the user logs in to node B, and the system can't find out about password "Z" since the network registries are not available.

However, if the user changes a password, then logs into node A at a time when the network registry is available, the system will update node A's local registry, and the account record will contain the password "Z."

Network Registry Objects

The following section describes the structure of the network registry. The network registry is a distributed, replicated database that allows a user to access the network under normal conditions. The network registry has the following structure:

- Master Registry File (one per network)
- Site Directories (one or more per network)
- Registry File Copy (one per node). Table 4-1 lists the locations and standard names for the network registry files and directories.

Table 4-1. Registry Object Standard Names and Distribution

Object	Standard Name	Data Location and Number
Master Registry File	*//node_name/registry/rgy_master*	Only one per net work
Site Directory	*//node_name/registry/rgy_site[n]*	At least one per network
Registry File Copy	*//node_name/registry/registry*	One on each node in network

We strongly recommend that you use standard names for these objects even though */registry/registry* is the only required name for correct operation of the registry. Append a name or number to */rgy_site* to differentiate site directories. Figure 4-6 illustrates the distribution of registry objects in a small network.

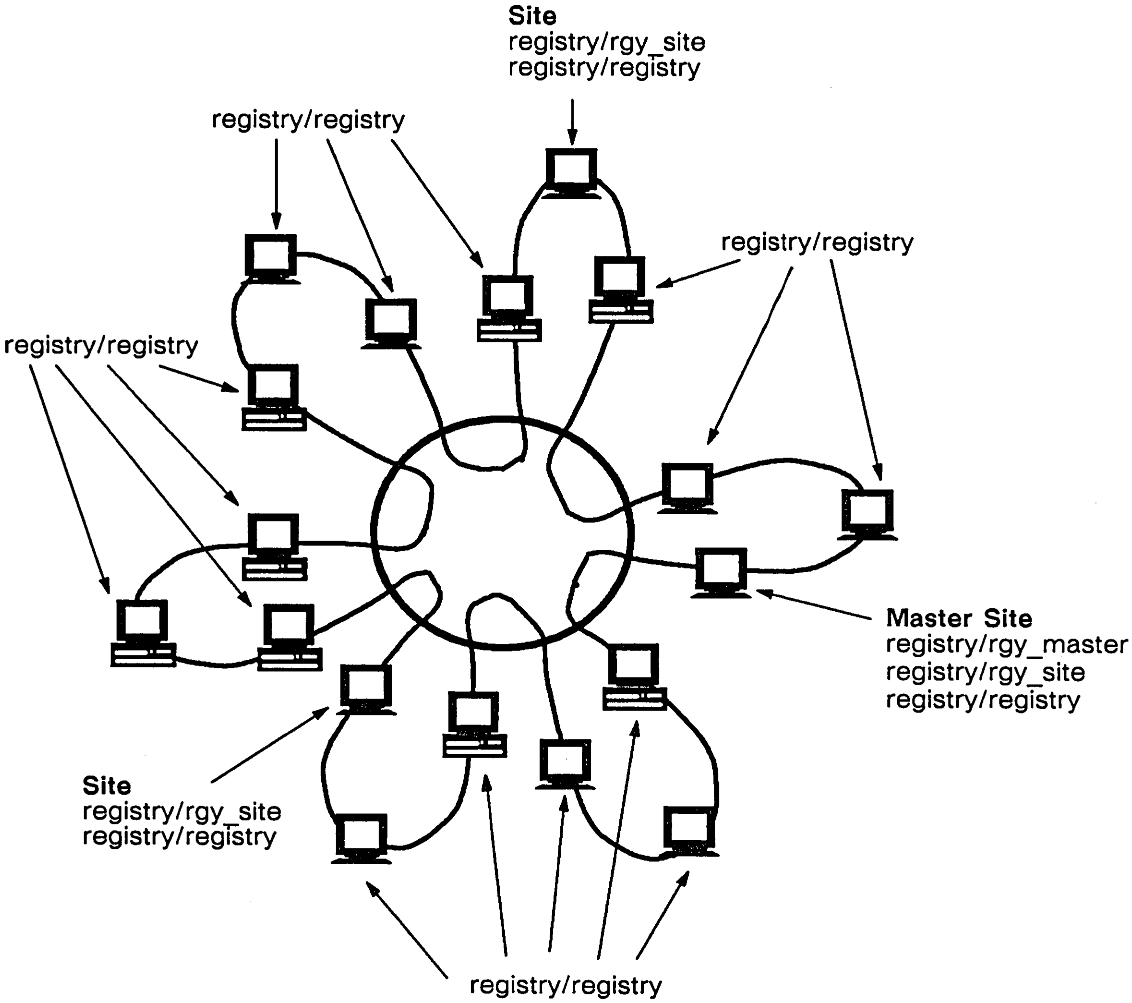


Figure 4–6. Distribution of Registry Objects in a Network

The Site Directory and Associated Data Files

Each **site directory** contains a copy of the database, the files that list authorized network users. Each node that has a site directory is a **site node** or simply a **site**. All site directories are identical. Each */registry/rgy_site[n]* directory contains five data files; **person, project, org, account,** and **full_names**. These data files identify the people, projects, and organizations that may use the network. Figure 4–7 illustrates the contents of these files.

/registry/rgy_site

person	**project**	**org**	**full_names**
name1name-id1	proj1proj-id1	org1org-id1	full name 1name2
name-id2	proj2proj-id2	org2org-id2	full name 2
:	:	:	:

account

name-id1	proj-id1	org-id1	password /home_directory
name-id2	proj-id2	org-id2	password /home_directory
name-id2	proj-id1	org-id2	password /home_directory
name-id3	proj-id1	org-id1	password /home_directory

Figure 4–7. Registry Data Files

The **person, project,** and **org** files contain 1 to 32 character string names and corresponding, unique, identification numbers. The system builds user accounts from these representations. Because each name has its own number, the same string name could represent a person, project, and an organization. For example, you could have person: Jones, project: Jones, and org: Jones. However, we recommend that you use different names for clarity. The registry also provides a **full_names** file that (optionally) associates a 32 character string with any name in the **person, project,** or **org** files. Typically, this file is used to associate a user's full name with the **ppo** names.

The registry **account** file contains records that associate one entry each from the **person, project,** and **org** files with a password and home directory. The order of records in the **account** file is significant. The system uses the first record it finds that matches the information given by the user to the log in prompt. A user may have accounts for two projects and want to log in to one of them by default. Manipulate the order of entries in the **account** file to achieve the result desired.

For example, user John Jones has project accounts ''widgets'' and ''washers'' and he wants to log in to his ''widgets'' account by default. Manipulate the order of entries in the **account** file as follows:

```
jones    widgets  none    password          //sam_node/jones/widgets
jones    washers  none    password          //sam_node/jones/washers
```

Now, if Jones logs in as ''jones,'' he gets the first matching identity, jones.widgets.none. He logs in as "jones.washers" to get his second account.

Enter names in the **person, project,** and **org** files with the **edppo** shell command. Enter accounts in the **account** file with the **edacct** shell command. Figure 4–10 shows examples of command usage.

There are default names in the **ppo** files that are useful in system management. Their specific purpose is discussed below. Table 4–2 shows the default names in the **person, project,** and **org** files.

Table 4-2. Default ppo Names

person	project	org
root	backup	apollo
sys_person	locksmith	none
user	login	sys_org
	none	
	server	
	sys_admin	
	sys_proj	

There are also default accounts which come with the system and these are shown in Table 4-3.

Table 4-3. Default account Names

user	none	none	/
user	sys_admin	none	/
user	backup	none	/
user	none	apollo	/
user	locksmith	none	/
user	server	none	/

The default **ppo** names and **accounts** are expected by certain system operations. For example, server processes created with the DM's **cps** command receive the name ''user.none.none". The project name *login* belongs to the ''login'' protected subsystem, described in Chapter 5. The person name *root* belongs to DOMAIN/IX.

The **ppo** names, *sys_person*, *sys_project* and *sys_org* can be used as required by the needs of your organization. The organization name *apollo* is for service representatives who log in to your system.

The **project** names *sys_admin*, *backup* and *locksmith* are for system administration purposes. The *sys_admin* account is general, the *backup* account is specifically for backing up system and users files, and the *locksmith* account can override all ACL rights assigned to any file or process. The account ''user.none.none" is the system default log in account.

Note that passwords for these accounts are not supplied. To further limit system access, assign passwords to all of the default system administration accounts. The ''user.none.none" account is discussed in Chapter 5.

The Master Registry File

One node in the network contains the master registry file. Any node may contain this file, but we recommend you place it on the node that contains one of the registry site directories. This node is called the **master site node**.

Each Node's Registry File Copy

The node's registry file copy is a copy of the master registry file. Every node in the network, including the node holding the master registry file, must contain a registry file copy.

Each Node's */registry* Directory

Each node's */registry* directory contains the registry file copy, a local registry file and local registry site directory. The DOMAIN/IX shell command **ls** lists the contents of this directory.

```
% ls /registry<RETURN>
Directory "/registry":
local_registry     local_site        registry rgy_siten
```

Figure 4-8. A Node's /registry Directory

On site nodes, the */registry* directory contains the subdirectory */rgy_site[n]*. The contents of */rgy_site[n]* are displayed in Figure 4-9.

```
% ls /registry/rgy_siten<RETURN>
account   full_names         org       person    project
```

Figure 4-9. Example of A rgy_site Sub-Directory

Creating and Maintaining Registries

This section reviews the commands used to create and maintain registries and gives general information about their use. Below, you will find described procedures used to create and maintain registries.

Shell Commands for Managing Registries

Table 4-4 lists the /**com** commands that create, update, and maintain the registry database.

Table 4-4. Registry Administration Commands

Descriptive Name	Command	Purpose
CREATE_REGISTRY	**crrgy**	Creates a registry that includes a master file of site pathnames, any number of site directories and person, project, organization, full_name, and account files. These files contain default names and accounts. Use the command to create local registries or to change registry sites. You must have write access to the registry data base and its directories to use this command.
EDIT_PPO	**edppo**	Creates, edits, and lists the names and associated full name text strings in the **ppo** files. Use **edppo** as a one-line command or as an

		interactive editor. You must have write access to the registry database to edit **ppo** files.
EDIT_ACCOUNT	**edacct**	Defines, changes, and lists accounts. Use it as a one-line command or as an interactive editor. You must have write access to the registry database to edit accounts.
LIST_REGISTRY	**lrgy**	Lists the contents of a registry file including, the pathname of the registry master file and the pathnames of all registry sites.
SALVAGE_REGISTRY	**salrgy**	Salvages a registry. Use **salrgy** to ensure that all sites of a registry contain the same information. You must have write access to the registry database to use this command.

Although all registry shell commands take a –r option that operates on a specific registry pathname, we strongly recommend that you not use this option. Instead, allow these commands to use the default, i.e., to use */registry/registry* to locate the master file and site directories. The one exception to this recommendation is for using the **lrgy** command to examine the contents of the */registry/registry* file on a given node. In this case invoke:

```
% lrgy –r //node_name<RETURN>
```

or, to examine the local registry on a given node:

```
% lrgy –r //node_name –loc <RETURN>
```

Creating Site Directory and Master Registry Files

When you create a registry for your network, execute Procedure 4–1 at the node chosen to contain the registry master file (*/registry/rgy_master*), and a registry site directory (*/registry/rgy_site[n]*). This node is called the **master site node**. Procedure 4–1 uses a shell script to create the following objects at site nodes:

- */registry* directory
- Master registry file, called *rgy_master* and registry site directory, called *rgy_site[n]*
- A copy of the registry master file, called */registry/registry*
- A local registry

After you create the *rgy_site* directories, use **edppo** and **edacct** in their interactive modes to enter names in the person, project, organization, full_names, and account files. As a last step, use another shell script to replicate the registry on the other nodes selected as sites. Procedure 4–2 creates the necessary registry files and directories on all the nonsite nodes in the network.

Protect your network registry files and directories from unauthorized access as soon as you create the registry with Procedures 4–1 and 4–2. Before executing Procedures 4–1 and 4–2, read the information in Chapter 5 on protecting system software and registries. Then execute the protection procedures immediately after creating the registries.

Procedure 4–1: Creating the Registry Database on the Site Nodes

Note: The */install* directory must be available to complete this procedure.

Execute this procedure on the master site node. Ensure that all the nodes are cataloged before executing this procedure.

1. At the master site node, run the **init_master_rgy** shell script in the installation utility directory */install*. Type:

```
% /install/init_master_rgy<RETURN>
```

2. Create */registry* directories at other site nodes, and add the site pathnames to the registry. For *//site_node_name_1,2...*, substitute the names of the nodes that you have chosen as site nodes. Type:

```
% mkdir //site_node_name_1/registry<RETURN>
% mkdir //site_node_name_2/registry<RETURN>
% mkdir //site_node_name_3/registry<RETURN>
% crrgy -a  //site_node_name_1/registry/rgy_site1<RETURN>
% crrgy -a  //site_node_name_2/registry/rgy_site2<RETURN>
% crrgy -a  //site_node_name_3/registry/rgy_site3<RETURN>
```

3. Use **edppo** to put names in the **ppo** files. **Edppo** accepts one to 32 character string names, and 32 character strings for associated full name text. To add a name, and optional fullname text type:

```
% edppo -a name [fullname]<RETURN>
```

To add a project, type:

```
% edppo -proj -a project_name<RETURN>
```

To add an organization, type:

```
% edppo -org -a organization_name<RETURN>
```

The example at the end of this section shows **edppo**'s interactive mode. It is more suitable than the command line when adding multiple names.

4. Create user accounts and assign passwords and home directories with the **edacct** command.

```
% edacct -a pers proj org homedir password<RETURN>
```

In interactive mode, **edacct** prompts for each account's **ppo** names, assigned password and home directory. The example at the end of this section shows **edacct**'s interactive mode. It is more suitable than the command line when entering multiple accounts.

5. Create a registry file copy and local registry for the site nodes, using the **init_rgy** shell script in the installation utility directory */install*. Specify the master site node's entry directory name, and the site node names, with the double slash (//):

```
% /install/init_rgy //master_site_node_name  //site_node_name<RETURN>
```

6. Read the information on protecting registries in Chapter 5. Then use Procedure 4–1 to protect the registry database you have created on each of these site nodes.

END OF PROCEDURE 4–1

Procedure 4–2: Creating Node and Local Directories

Create the remainder of the registry database, on nodes that are not site nodes. Use this procedure at every nonsite node in the network. Do the procedure after Procedure 4–1.

1. At every nonsite node in the network, create a registry directory, registry file copy, and local registry with the **init_rgy** shell script in the installation utility directory, */install*. Specify the master site node's entry directory name, with the double slash (//) before it:

   ```
   % /install/init_rgy //master_site_node_name<RETURN>
   ```

2. Use Procedure 4–1 to protect the registry files and directories on each node.

END OF PROCEDURE 4–2

Figure 4–10 shows a sample session, using the interactive forms of the **edppo** and **edacct** shell commands to define **ppo** names and create user accounts in a new registry.

```
% edppo => l                                            # List the names in
                                                        # rgy_site


  root   sys_person   user
=> a                                                    # Add new names
add=> jones 'John J. Jones'
add=> donnell 'Rose Donnell'
add=> sas 'Susan Smith'
add=> jih
Enter full name in quotes: 'James I. Harris'            # prompts for full name
add=> pjk
Enter full name in quotes:<RETURN>                      # full name is optional
add=><RETURN>                                           # to end adding names
=> l                                                    # List the names in
                                                        # rgy_site
jih    jones        donnell     pjk    root
sas    sys_person   user
=> lf                                                   # list with full names
jih               James I. Harris
jones             John J. Jones
donnell           Rose Donnell
pjk
root
sas               Susan Smith
sys_person
user

#  Up to this point no information has been written to rgy_site.
#  The next step is critical because we actually write out the changes using the
#  "wr" interactive command.

=> wr                                                   # update the file and
                                                        # quit
% edppo -proj                                           # define two projects
=> a finance 'finance'
=> a mkte 'Marketing, East Coast'
=> wr                                                   # update the file and
                                                        # quit
% edppo -org -a ajax_distrib                            # define one org as a one
                                                        # line command
```

Figure 4–10. Sample Session

```
% edacct                                                # define some accounts
                                                        # using the ppo names
=> la                                                   # list all entries
user     none      none      /
user     sys_admin none      /
user     backup    none      /
user     none      apollo    /
user     locksmith none      /
user     server    none      /
=> t                                                    # go to top of file
=> a jih finance none //fin/jih ' '                     # no password
=> a jones mkte none //mkt/jones 'john'                 # assigned password
=> a donnell  none none //od/rose ' '
=> a donnell  sys_admin none /  ' '
=> ln                                                   # list next entry
user          none          none          /
=> lc                                                   # list new or changed
                                                        # entries
jih           finance       none          //fin/jih
jones         mkte          none          //mkt/jones
donnell       none          none          //od/rose
=> wr                                                   # update the file and
                                                        # quit
# we now have the following accounts in this registry:
% edacct -l
jih           finance       none          //fin/jih
jones         mkte          none          //mkt/jones
donnell       none          none          //od/rose
user          none          none          /
user          sys_admin     none          /
user          backup        none          /
user          none          apollo        /
user          locksmith     none          /
user          server        none          /
```

Figure 4–10. Sample Session (cont.)

Maintaining Registries in Existing Networks

You are responsible for maintaining and protecting your registry database. This section provides guidelines for registry maintenance and protection. See Table 4-4 for a summary of the commands used to update the user account database and to add and delete registry sites.

To add and delete registry sites, use the **/com/crrgy** command with the **-a** or **-d** option. **Crrgy** adds or deletes site names to the master registry file depending upon which option you use. **Crrgy** also has the capability of adding sites to the *rgy_master* file without accessing any of the sites (**-exist** option). This option can be useful when you are preparing to create a site on a node that is not yet available. The **-d** site option never accesses the deleted sites (in case they are unavailable when the command is issued). Instead, **crrgy -d site** deletes the site name from the *rgy_master* file. Refer to the *DOMAIN System Command Reference* for the **crrgy** options.

If you execute the INVOL utility on a site node, or change the node's name, or replace its disk, follow the procedures in Chapter 2 for maintaining root directories. Use the procedures described for **ctnode** or **ns_helper,** depending on how you have chosen to maintain root directories at your site. A registry is an upper-level directory, and as such, is subject to the rules of the network naming structure. Maintain site node entry directories as you would any other entry directory.

Procedure 4-3: Adding A Registry Site

1. Add a registry site by executing the following command at the master site; type:

   ```
   % crrgy -a //new_node/registry/rgy_site[n]<RETURN>
   ```

   This command will create a new registry site at *new_node* and update the file */registry/rgy_master*. It does not update */registry/registry* anywhere in the network.

2. After you have added the new registry site update all node /registry/registry copies of the master file */rgy_master*. Type:

   ```
   % cp  /registry/rgy_master //alpha/registry/registry<RETURN>
   % cp /registry/rgy_master //beta/registry/registry<RETURN>
   ```

END OF PROCEDURE 4-3

Procedure 4-4: Deleting A Registry Site

1. Delete a registry site by executing the following command at the master site; type:

   ```
   % crrgy -d //node/registry/rgy_site[n]<RETURN>
   ```

   This command deletes only the pathname *//node/registry/rgy_site[n]* from the file */registry/rgy_master*. It does not delete the directory *//node/registry/rgy_site[n]*.

2. After you have deleted a registry pathname from *rgy_master*, delete the directory *rgy_site*.

   ```
   % rm -r //node/registry/rgy_site[n]<RETURN>
   ```

3. Update all node */registry/registry* copies of the master file *rgy_master*. Type:

   ```
   % cp  /registry/rgy_master //alpha/registry/registry<RETURN>
   % cp /registry/rgy_master //beta/registry/registry<RETURN>
   ```

END OF PROCEDURE 4-4

Procedure 4-5: Copying Replacement Master Registry

This procedure does not create any new sites. This example assumes you are using the same sites, and moving only the file */registry/rgy_master*.

1. Create a replacement master registry file on this node. Type the command shown on a single shell command line:

   ```
   % crrgy -ex -s //node/registry/rgy_site1 //node/registry/rgy_site2<RETURN>
   ```

2. Update all node */registry/registry* copies of the master file *rgy_master*. Type:

   ```
   % cp /registry/rgy_master //alpha/registry/registry<RETURN>
   % cp /registry/rgy_master //beta/registry/registry<RETURN>
   ```

END OF PROCEDURE 4-5

Procedure 4-6: Moving Master Registry

The following commands issued from *//newmstr* will move a registry from *//oldmstr* to *//newmstr* and add two additional sites.

1. Create new *rgy_master* file without creating data files. Type the command shown on a single shell command line:

   ```
   % crrgy -r //nmstr exists -s //newmstr/registry/newsite1  //node2/registry/newsite2
   //node3/registry/newsite3<RETURN>
   ```

2. Move existing data files to the first new site. Type the command shown on a single shell command line:

   ```
   % cp //oldmstr/registry/oldsite1 //newmstr/registry/newsite1<RETURN>
   ```

3. From the *//newmstr* node, replicate *rgy_master* on *//newmstr*. Type:

   ```
   % cp /registry/rgy_master /registry/registry<RETURN>
   ```

4. From the *//newmstr* node, move the data files to other new sites. Type:

   ```
   % salrgy<RETURN>
   ```

5. Distribute the new copy of *rgy_master* to every node in the ring. Type:

   ```
   % cp //newmstr/registry/rgy_master //?*/registry/registry<RETURN>
   ```

   Note that the registry files at the old sites remain untouched.

   There may be difficulties doing this procedure if nodes are unavailable. Nodes will need to be updated when they are back on the ring.

   Additional difficulties may be encountered due to ACLs either on *//newmstr* or on other nodes. The registry system cannot violate ACL protections and it is the system administrator's responsibility to ensure that ACLs are correct for correct operation of the system.

END OF PROCEDURE 4-6

Maintaining Registry Database Consistency

To monitor and maintain registry sites, use **/com/lrgy** (LIST_REGISTRY) and **/com/salrgy** (SALVAGE_REGISTRY). The system automatically verifies database consistency whenever you or other users write to the registry. (You write to the registry when you change it using **edacct** or **edppo.** Users write to it during log in if they change their passwords or home directories.) The system performs its own version of **salrgy** if it finds database discrepancies. Execute **salrgy**

- whenever you are unsure of registry database consistency
- whenever a registry site which was not on the network during a network registry update comes back to the network.

Maintaining the Database

Network configuration changes may require that you move the registry database locations. Staff and security changes will require that you add, delete, and change user accounts. Use registry administration commands to update and maintain the registry database. Table 4-4 summarizes these commands. See the *DOMAIN System Command Reference*, or on line HELP files, for detailed descriptions of each command.

To change the registry database, use **edppo** and **edacct**. Add and delete names in the **ppo** files, and change users' accounts, home directories or passwords as necessary. Note that the master site, and all other registry sites, must be available when you use **edppo** or **edacct**. If the sites are unavailable, the system gives an error message stating that the registry update is only partially complete. If you edit **ppo** or **account** files after receiving this message, execute **sarlgy** when the sites are back in the network. Also, back up the registry files promptly whenever you change the user account database.

You can change any password with **edacct** even if you do not know the account's current password. Users may change their passwords and home directories at log in. These changes are recorded automatically in the **account** file. If some registry sites are unavailable, users also receive the "partial update" error message described above. See the *DOMAIN System User's Guide* for information about changing passwords or home directories at log in. The system maintains passwords in one-way encryption after they are entered in the **account** file. If a user forgets a password, a system administrator can assign a new one with **edacct**.

The shell command **/com/chpass**, (CHANGE_PASSWORD) changes passwords from the command line. **Chpass, edacct, edppo,** and several other shell commands belong to a structure called the "login" protected subsystem. Chapter 5 discusses protected subsystems in detail.

Adding New Nodes to the Network

Whenever you add a new node to the network, first catalog the node as described in Chapter 2. Then use Procedure 4-2 to copy the registry files and directories to the node.

Setting ACLs on netsvc

The default ACL for **netsvc** is shown in Figure 4-11. It gives execute rights (x) to any user.

```
% acl /com/netsvc -all
Acl for /com/netsvc:
  %.%.%.%                     pgndwrx
```

Figure 4-11. ACLs on netsvc

You can change the ACL on **netsvc** on nodes containing registries so that only a *sys_admin* account can have execute rights. The *DOMAIN System User's Guide* describes how to set ACL entries. This will protect the registry from careless or naive users working directly at thatas it is node.

The DOMAIN/IX passwd and group files

The DOMAIN/IX system includes the file */etc/passwd*, an analog to the standard UNIX password file, and the file */etc/group*, which is similar to the UNIX group file. The structure of these files, and a full description of their contents, is included in the *DOMAIN/IX Command Reference* pages for **passwd**(5) and **group**(5).

The *passwd* and *group* files exist to provide user account and group information in a form familiar to DOMAIN/IX system users and DOMAIN/IX programs. The files contain information from the DOMAIN network registry's **ppo** file and the site directory's **acct** file, which have been placed in the DOMAIN/IX format. You must use the **crpasswd**(8) command to change the contents of either of these files, with one exception that is explained below.

> **Note:** As of SR9.2, all DOMAIN network registry information must appear in the correct case. If, for example, a user's home directory is listed in the registries, and thus in */etc/passwd*, as *//FOO/BAR*, even though the actual directory name is *//foo/bar*, DOMAIN/IX programs that get this information from */etc/passwd* will not be able to find the directory.

The passwd file

The DOMAIN/IX *passwd* file contains the following information for each user's account:

- log in name
- a password field (always empty)
- numerical user ID
- numerical group ID
- full name of user and uid
- initial working directory
- program to use as shell

As we said above, the *passwd* file exists on DOMAIN systems to provide user account information in a form familiar to DOMAIN/IX users and a form that DOMAIN/IX programs can use. The file does not include passwords at all and the operating system does not use it to verify a user's identity at log in time. A typical entry in the *passwd file* might look like this:

```
lulu::2775:38:Patti Labelle, 12345678.323://rock/lulu:/bin/csh
```

Note that the second field, which on DOMAIN/IX systems is the field that contains the user's encrypted password, is null. On DOMAIN systems, all passwords are maintained by the registry, and are not copied into the *passwd* file.

The only field you may edit by hand in the *passwd* file is the last field, the one which controls the 'program to use as shell'. In our sample entry, the C shell (*/bin/csh*) is the program that user 'lulu' wishes to use as shell. If you wanted to change that field, you would edit the *passwd* file manually. Otherwise, you should not edit the *passwd* file. If you do change the shell field, run **/etc/crpasswd** immediately afterward. If you are adding a new user to a DOMAIN/IX system, use the appropriate procedures in this chapter for adding a new account to the registry, update the 'shell' field for that account in */etc/passwd*, if necessary, then run **/etc/crpasswd**. More information about **crpasswd** is included below.

The group file

The DOMAIN/IX *group* file contains the following information for each user's account:

- group name
- a password field (always empty)
- numerical group ID
- a list of the log in names of all members of the group, with names separated by a comma

Like the *passwd* file, the *group* file exists on DOMAIN/IX systems only to provide user account information in a form useable to DOMAIN/IX programs and users. It is an ASCII file which is created and updated by **crpasswd**; you should not edit this file, either. Like the *passwd* file also, nothing is placed in the password field; DOMAIN/IX does not support group passwords. A typical entry in the *group* file looks like this:

```
backup::1:lulu,huey,neil,eric,robbie:
```

A user may specify a default group, but his or her log in name will only appear in the *group* file entry for groups other than the default.

The passwd.map file

The file named */etc/passwd.map* is used by the system calls **getuid** and **getgid** to translate DOMAIN subject identifiers to user and group IDs, and vice versa. For more information about the role of the */etc/passwd.map* file in translating DOMAIN/IX modes to DOMAIN ACLs, and vice versa, see the section in Chapter 5 which explains modes and ACLs.

The crpasswd Command

The **crpasswd** command creates the necessary password and group files (*/etc/passwd*, */etc/passwd.map*, and */etc/group*) from the DOMAIN network registry's person, project, and organization (**ppo**) files and the site directory's (**acct**) file. Running **crpasswd** is the only way you should alter any of these files, except in the case we mentioned above, when changing the shell field in */etc/passwd*. (This 'shell' field cannot be derived from information in the registries.) Always run **crpasswd** after changing the shell field. **Crpasswd** also updates the */etc/passwd.map* file.

Note that the **cron** program runs with a 'root' identifier, so that you can have **cron** run **crpasswd** daily, to keep the password and group files up to date. Most sites use **cron** to run **crpasswd** at least once a day. See the section on **cron** in Chapter 6. To invoke **crpasswd** from a DOMAIN/IX shell, you must be logged in as super-user (or root).

```
% /etc/crpasswd
```

The command will print out lines as it moves through various stages of operation:

```
Reading /etc/passwd.
Reading /etc/group.
Updating /etc/* from registry project file
Updating /etc/* from registry person file.
Updating /etc/* from registry account file.
Writing /etc/group.
Writing /etc/passwd.
Writing /etc/passwd.map.
```

The /etc/crpasswd program is sufficiently intelligent to update the */etc/passwd* and */etc/group* files. Do not delete these files before running **crpasswd**.

If you do delete */etc/passwd*, DOMAIN/IX **userids** assigned by a later invocation of **crpasswd** may not map correctly to system UID's held in the *'node_data/acl_cache*. One result of this is that the **chmod** command will change the owner of a file, as well as the access mode. If you accidentally delete */etc/passwd*:

- Clear each node's *'node_data/acl_cache*, preferably by running /etc/**flush_cache** on every node, disked or diskless, in the network.

- Run **crpasswd**.

For more information on the *acl_cache*, see Chapter 6.

Chapter 5

Protecting System Registries and Software

The network registry allows the operating system to control user access to the network. You will use the methods described in this chapter to protect system and user software on each node. You will also perform regular backups of system and user directories to protect work in progress. Use subsystems, ACLs and SIDs to provide selective levels of security for parts of the system or directories. This chapter describes:

- Protecting registries and system software on nodes
- Installing new DOMAIN software
- Creating system backups to protect user files
- Using subsystems, SIDs and ACLs to selectively grant access to the node or network.
- The relationship between ACLs and DOMAIN/IX modes, the concept of ''DOMAIN/IX ACLs'', and the *acl_cache*.

We assume that you are familiar with the concepts of protected subsystems, ACLs and SIDs. If these concepts are unfamiliar to you, read about them in the *DOMAIN System User's Guide* before attempting to master the material in this chapter.

Protecting Registries and System Software

Every object in the system has an ACL that specifies the rights (read, write, delete, etc.) a user has to that object. The *DOMAIN System User's Guide* explains how to create ACLs. As described in the previous chapter, secure networks contain registries that control users' access to network resources. In an open network, one without registries, every user has complete rights to every object. When you receive new nodes with disks, the ACLs for objects on the disk make it an open network node.

After you create a registry, the network will not be secured entirely until you execute a **protection program** supplied with the DOMAIN system software. The program protects system software and all parts of the registry residing on each node. The program simplifies the task of assigning a complex set of ACLs at each node. This section describes the levels of protection available for the nodes in the network. Procedure 5–1 describes how to execute the protection program.

Execute this program at each node when you first set up the network. The program changes the default ACLs on a node to be consistent with the level of protection you select. Whenever you add another disked node to the network, decide on the level of protection the node should have and execute the program. When you receive a new software release, the protection program is included automatically by the software installation procedures.

Selecting the Level of Protection

The protection program uses two files to protect system software, *sys_acls* and *sys_idacls*. The files are created in the utility directory, */install/acl_dir*. The contents of these files come from a set of **ACL templates** also supplied with your system. ACL templates are lists of ACL entries and initial, default ACL entries for system files and directories, including registry files and directories. The protection program reads *sys_acls* and *sys_idacls*, and assigns the ACLs listed in these files.

When you execute the protection programs on nodes with newly created registries, and on newly installed nodes, you select ACL templates. When you install a new DOMAIN software release, the software installation procedures automatically copy the ACL templates you select into the files.

ACL templates are based on the following assumptions about network users:

- General users simply use a node, and have no administrative privileges or responsibilities. ACLs can prevent these users from changing or deleting any system software.
- Node administrators are responsible for the proper operation of a particular node(s). ACLS can grant them access to libraries, the operating system, and local commands.
- System administrators are responsible for the overall security of a network. ACLs can grant them full access to all parts of the system including registries, protected subsystems, and system commands, that are not available to the node administrator.

ACL templates let you assign different sets of ACL entries to different nodes in the network. For example, you can arrange that on some nodes any user has access to system software, but on other nodes only system administrators have this access. For this reason, three categories of ACL templates are available. There are two templates in each category. One template assigns ACLs and the other assigns initial default ACLs. If the ACLs assigned by the templates do not match your needs, edit either *sys_acls* or *sys_idacls* to modify the ACLs to fit your requirements.

The ACL template categories create the following types of nodes:

1. *Open_node* has the lowest protection. Any user can perform node administrator functions, but a system administrator is required for some functions.
2. *Personal_node* has moderate protection. Node administrator privileges are limited to a single user or group of users. A system administrator is required to perform some functions.
3. *System_node* has the highest protection. Only system administrators can perform any administrative functions on this node.

Table 5-1 shows the pathnames of ACL templates in each of the three categories.

Table 5-1. ACL Template Filenames

Node Type	Template Filenames (all in install/acl_dir)
open_node	secure_net_open_node_acls secure_net_open_node_idacls
personal_node	secure_net_personal_node_acls secure_net_personal_node_idacls
system node	secure_net_system_node_acls secure_net_system_node_idacls

The *open_node* ACL templates offer minimum protection for a secure network. On nodes protected with this option, general users (including user.none.none) have the same rights to much of the system software as system administrators. These templates withhold rights only for self-protection, i.e., users can't delete system software or registries.

The *personal_node* ACL templates offer moderate protection for a secure network. In copies of these templates, you specify a node administrator who has more rights than general users, but fewer rights than system administrators. As with open node templates, the personal node templates prevent anyone from deleting pieces of system software, as protection for the network. Also, they allow only system administrators access to security-related files and directories.

The *system_node* ACL templates allow only system administrators access to system software. Of the three categories, these templates provide the greatest protection.

No matter what protection category you choose, you can make some kinds of safe modifications to your registry, yet maintain overall registry security. These modifications are described in the section on Protected Subsystems.

Reading ACL Templates

To see the entire set of ACLs the protection program assigns to nodes, display any of the ACL template files listed in Table 5-1 on a monitor screen. The template will serve as a reference when you read Table 5-2.

The top of each ACL template file contains some comment lines. Scroll through the comment lines to a table consisting of the pathnames of system files, directories, or wildcard specifications. The ACL entries assigned to the listed objects appear at the right. The pathname and ACL entry has the format shown in Table 5-2.

Table 5–2. Fields in ACL Templates

Column No.	Contents
1–28	Pathname of file or directory; must start with ′/′; may be wildcard
29	′f′ or ′d′ for File or Directory
33–40	Rights assigned to %.%.%.% (anyone) All rights in the table assigned in the form used by the edacl command (see the *Domain System Command Reference*).
45–52	Rights assigned to %.sys_admin.%.% (any system administrator)
57–64	Rights assigned to additional SID class (#1) (may be left blank). When you use the personal_node templates, these rights are for node owner, the person responsible for the node, and are already assigned.
69–76	Rights assigned to additional SID class (#2) (may be left blank or filled in). The header for this column is "PPO2."

The following line, from a *sys_acl* template in the *secure_net_personal_node* category, shows the ACL assignments for the */sys/help* directory.

```
                        % rts          %.sys_admin   PPO1

/sys/help               d   r          pgndcalr     pgndcalr

                        READ and
                        DELETE
                        rights for     All rights for %.sys_admin.%.%
                        %.%.%.%        and for the person responsible
                                       for this node.
```

If you execute the protection program with the *secure_net_personal_node* category, give the name of the node "owner" or administrator, for example, Fred. The ACL for the */sys/help* directory shows the result of this *sys_acl* specification. Fred.%.lab.% gives full rights to the node administrator and a system administrator, but not to any user.

```
% acl /sys/help
Acl for /sys/help:
   fred.%.lab.%                 pgndcalrse
   %.sys_admin.%.%              pgndcalrse
   %.%.%.%                      ---d---r
```

Editing ACL Templates

To edit a copy of ACL templates, the protection procedure directs you to copy the templates you choose into *sys_acls* and *sys_idacls*. Edit these files if you want to:

- Change the rights listed in the ACL templates
- Specify additional rights for SID classes 1, and 2 (if you chose *open_node* or *system_node* templates)
- Specify additional rights for SID class 2 (if you chose *personal_node* templates).

The following paragraphs provide some general guidelines for editing ACL templates.

When you edit ACL template copies, do not eliminate any of the rights assigned to *%.%.%.%* and *%.sys_admin.%.%* in the template. Eliminating any of these rights could affect the operation of your system software.

The bottom portion of an *open_node* template contains files and directories that affect network security. The rights specified in the template for these objects are the maximum allowed for secure system operation. Granting more rights than those specified can make your system insecure.

If you specify rights for each additional SID class listed in Table 5–2, be sure to use the correct columns. The template lists the column numbers.

Specify the same rights that you specify with the **edacl** command. For files, specify **pgndrwx** or a subset thereof (use '–' in place of any right). For directories, specify **pgndcalrse** or a subset thereof (use '–' in place of any right).

Whenever you specify your own name (as it appears in *rgy_site[n]/person*, in one of the additional SID classes, give yourself **p** rights. The ACL creation will fail if you don't authorize yourself to change ACL entries.

ACL templates do not set ACL entries for the node's // entry directory. Assign ACLs to node entry directories manually. Preface comments in the template file with the comment character, #. The protection program ignores the comment character and anything following it.

Running The Protection Program

This section provides a procedure for running a program that protects your registry database and system software in a secure network. Use Procedure 5–1 when you:

- Create a registry for the network
- Create a registry for a new node in an existing network
- Change the location of *rgy_master* or *rgy_site[n]* and the new location does not have adequate security.

When you install new software in a secure network, the software installation procedure automatically executes the protection program.

Run the protection procedure on any disked node in the network. Before executing the procedure, use the **lvolfs** command to determine the amount of disk space available on the node. The procedure needs 3000 blocks of disk space. If the disk space is inadequate, remove user files, not DOMAIN system files, to perform the procedure. When the procedure is finished, return files to the disk.

Run the procedure first on the master site node. Then, repeat it on each of the other site nodes, and finally on each nonsite node. The master site node must be available when you run the procedure on other nodes.

If a node's system software contains links to other system objects, the protection procedure sets the ACLs for the object to which the link points. For example, if */sys/help* on one node is a link to the */sys/help* tree on another node, the protection procedures will set ACLs for the second node's */sys/help* tree.

In a new network, execute the protection procedure immediately after creating the registry and before creating any links in place of standard software. Then, you need not be concerned with links when you run the protection procedure.

If there are links on nodes, of the type described above, delete the links before executing the procedure, then recreate them when the procedure is finished.

> NOTE: Before executing this procedure, the node must be catalogued and must have its registry files and directories (See Procedure 3–1 or 3–2.)

The next procedure illustrates responses to the protection program prompts. Give responses appropriate to your site when you execute this program. The output shown illustrates the "most complicated case" i.e., there are customized ACLs for the node. If you select an ACL template from the ones supplied, there are fewer prompts than are illustrated here.

Procedure 5-1: Procedure for Protecting the Registry Database

1. Log in with a *sys_admin* account. Set the working directory to */install* and invoke the install procedure:

```
% cd install
% install<RETURN>

Software Installation Types are:

    STD        -- Install SR9 standard software
    RESTART    -- Restart the software installation
    OPT        -- Install optional software
    ACL        -- Set acls for existing software
    CLEANUP    -- Run the cleanup procedure for ADD MODE installations
    DOMAIN/IX  -- Install the DOMAIN/IX software

Please enter Installation Type: acl<RETURN>
You are logged in as:
    person.sys_admin.organization.node  //node_name
Do you have adequate rights?
Please enter response. (yes or no) yes<RETURN>
Please enter //TARGET_NODE name: //george<RETURN>

ACL Template Types are:


    OPEN      --       This type of node has the lowest protection.  Any  user
                       can  perform  node  administrator functions. A  system
                       administrator  is  required  for  some functions.

    PERSONAL --        This type of node has moderate  protection.  Node admin-
                       istrator privileges are limited to a single  user or
                       group of users.  A system  administrator is required to
                       perform some functions.


    SYSTEM    --       This node has the highest protection. Only system admin-
                       istrators  can perform  any  administrative functions on
                       this node.


    USER --            Choose this option  if you  have  saved  your ACL tem-
                       plates from a previous installation or  update  and wish
                       to reinstall those ACLs.  You must  copy  your own cus-
                       tomized  version of the ACL  templates to the  files
                       //node_name/install/acl_dir/sys_acls and //node_name/in-
                       stall/acl_dir/sys_idacls before     you can continue.

Please enter the type of ACL template you would like: user<RETURN>

Have you already copied your customized version of  the ACL and IDACL tem-
plates  to the files:
    //george/install/acl_dir/sys_acls
    //george/install/acl_dir/sys_idacls
 Please enter 'YES' or 'NO' : yes<RETURN>
```

If you enter "NO," the script prompts as follows:

```
Please copy your own customized version of the ACL template  to the files
sys_acls and sys_idacls. Do this by typing:

% cp customized_acls //george/install/acl_dir/sys_acls
% cp customized_idacls //george/install/acl_dir/sys_idacls

 Then you must RE-START this procedure.
```

If you have already copied your customize ACLs to the target node the script will continue as follows:

```
Have you specified an additional PPO?
Please enter 'YES' or 'NO': yes<RETURN>
```

If you answer "YES" the script will prompt you for the PPOs as follows:

```
Please enter the first PPO: #enter additional PPOs here#

   PREPARING ACLS
          .
          .
          .
  SETTING ACL'S
  ACL'S CAN NOT BE SET FOR OBJECTS THAT ARE IN-USE.

  NOTE: You may get the following error message:

 ?(acl) Acl not changed for   - object is in use (OS/file server)
 This error is expected and will not affect the installation procedures.
          .
          .
          .
    Finished Installing Acl's on //george
    Please shutdown, reset and restart //george
```

2. Shutdown, reset and restart the node.

Installing New DOMAIN Releases on Secured Networks

We only guarantee that consecutive software releases are compatible. Therefore, do not try to run a network with some nodes running SR7.0 or SR8.0 software and some running SR9.0 software. Software installation instructions are included in the release notes.

Only persons with a *sys_admin* account can initially install software. Refer to the *SR9 Release Document* for the procedures to use to install new software on nodes. The software installation program uses the protection program, described previously, to protect the new software and registry.

Backing Up Your System

To protect the information on nodes, copy files to some off-network storage media, such as magnetic tape, at regularly scheduled intervals. It is not necessary to copy every file, on every node, to tape. Save system software once. If you do not want to back up copies of system software at all, it is especially important to save the media shipped with the nodes.

Since many nodes in a network hold the same system software, be selective about the files that are backed up regularly. Each node contains, for example, the same */systest* directory. Save this directory from one node and restore it to other nodes as necessary.

One way to selectively save files is to create a back-up list on every node. Tell users to enter the names of upper-level directories that they want to save in the node's backup list. Save system and server log files by entering their names in the back-up list. Then write a shell script that reads the back-up list and writes the directories in the list to magnetic tape. Use the **wbak** (WRITE_BACKUP) command to write the files to magnetic tape.

Remember that the system cannot write a file to tape unless the Access Control List (ACL) for the file allows READ access. If you write a shell script to create the system backups, be sure that your system's ACLs are such that the shell program can read all needed files.

Wbak creates a file called *backup_history* in each directory that it backs up. The log in account that executes **wbak**, directly or through a shell script, must have ADD access to the directories being backed up and WRITE access to the *backup_history* file. See below for detailed information about the *backup* project account for people doing system backups.

You can restore the objects saved on tape to a destination directory with the **rbak** (READ_BACKUP) command. By default, **rbak** assigns the destination directory's default ACL to the object being restored. So, by default, the restored file or directory has the same access control assignments as the directory to which it is restored. If you want the object to retain its original ACL, you must use the **-sacl** option with **rbak**.

Protected Subsystem Status

The operating system associates a *subject identifier* (SID) with each process the user creates from log in to log out. For example, processes executed by the user's start-up file have SIDs. SID classes correspond to the registry **ppo** files. ACL rights are attached to SID classes.

The system gathers SID information during the log in procedure. The first field in an SID contains a person name, the second field is the user's project name, the third field contains organization names. The information contained in these three fields is referred to as an SID class. The fourth field in an SID identifies the hexadecimal ID of the node where the user is working. The system assigns the same SID to every process a user creates during a log in session.

An exception to the usual procedure described above occurs when the user executes the **/com/login** command in the shell input window. In this case, the operating system starts a new log in process and assigns a new SID to that process. The SID for this process also ends when the user logs off.

When a process requests access to a file, directory, program or another process, the system compares the process's SID with the ACL of the requested object. If the requested object's ACL includes an SID specification that matches the process's SID, the system grants the process the access rights specified in the ACL.

Usually, a program can access a file if the process running the program has the proper SID. The ACL templates supplied with your DOMAIN system insure that system server processes have SIDs and ACLs that will not hinder user access.

At times, you may want to grant a program access rights to certain data files, regardless of the program's process SID. The term "protected subsystem" refers to a set of data files and a program that has special access rights to the files. The *DOMAIN System User's Guide* describes how to create protected subsystems. In the following sections we describe those aspects of protected subsystems of particular interest to system administrators.

The LOGIN Protected Subsystem

The protection program makes your registry a data object in the "login protected subsystem." The login protected subsystem allows users to change their own passwords or home directories while logged in, or to change SIDs while logged in. However, only system administrators can add or delete accounts, create new site nodes, or other operations that affect the security of the network at large.

The following list shows the shell commands that are part of the login subsystem.

- **Chpass** allows users to change their password at the command shell level.
- **Chdir** allows users to change their home directories at the command shell level.
- **Edacct** allows system administrators to edit the **account** files.
- **Eddppo** allows system administrators to edit the **person, project,** and **organization** files (see Chapter 4).
- **Xsubs** allows shell programs to run as subsystem managers.
- **/Com/login** allows users to change their SIDs. Using this command, users can log in to an existing process with a different SID. For instance, a user who originally logged in with the SID "nicholas.none.none.1cad" might need the more specific SID "nicholas.none.dickens.1cad" to have full rights to an object.
- **Siologin** is part of the login subsystem that allows users to log in to a DOMAIN network via a modem or terminal attached to a serial I/O (SIO) line on a DOMAIN node. Chapter 6 describes **siologin.**

Refer to the *DOMAIN System Command Reference* for information about the shell commands listed above. The system runs several other login manager programs for use with DSPs.

These are the default ACLs on the login protected subsystem.

```
% acl /sys/subsys/login
 Acl for sys/subsys/login:
 Subsystem login manager
   %.sys_admin.%.%                  p-n--rx
   %.%.%.%                          -----rx
```

Using Protected Subsystems to Grant Access Selectively

This section provides an example of how one system administrator used ACL entries and protected subsystems to protect system resources. The example shows how administration personnel at one site protected a set of node listing records.

The example that follows is very simple, but there are more complex and powerful uses for protected subsystems. If you have sensitive programs that you want users to run only with certain options, you could write manager programs that run these programs, and deny EXECUTE rights to the sensitive programs themselves. You could do the same thing if you only want the sensitive programs run for certain purposes.

In our example, a system administrator's assistant, named Art, used the Netmain Interactive Tool to produce a weekly listing of all the nodes in the network. He named the file *node_list.date* (in yy.mm.dd format), and stored it in his *records* directory, a subdirectory of his home directory. Fran, the system administrator, wanted Art to sort some of the information and store the sorted files in her *confidential* directory. She did not want Art to have read or write rights to any of the files in the *confidential* directory. The following is a description of how she set up ACLs for these files and directories, including a transcript, shown in Figure 5-1.

Fran first created a protected subsystem called *node_list_protect*. She then created several links to her confidential directory, and created a dummy file in this directory. She defined the dummy file ACL entries so that she was the only user with full rights, and made it a *node_list_protect* data file. Then she wrote the file-sorting shell script, and put a copy in Art's personal *com* directory.

The script starts with the **subs -up** command to increase privilege to allow it to act on subsystem objects. The script then creates several files. (The script also includes a command that sets ACLs to make these files into *node_list_protect* data files.) Finally, Fran made the script a *node_list_protect* subsystem manager. When Art wanted to use the sort program, he executed the following command:

% xsubs node_list_protect yy.mm.dd<RETURN>

Figure 5-1 shows the transcript.

```
      #Create the subsystem.
 % crsubs node_list_protect

    #Create home directory links to
    #//mc/fran/confidential.
    #Use links instead of the meaningful, but cumbersome, pathname.

 % crl ~con //mc/fran/confidential
 % cpl ~con //mc/art

    #Create a dummy file in ~con.
% wd ~con
% crf node_list.dummy

    #Enter the node_list_protect subsystem.
% ensubs node_list_protect

    # Give node_list.dummy data object status in node_list_protect.
% subs ~con/node_list.dummy node_list_protect -data

    # Leave the protected subsystem.
% *** EOF ***

    # Make ACL entries for node_list.dummy.
 % edacl ~con/node_list.dummy -cf fran.%.lab.% -owner
 % edacl ~con/node_list.dummy -cf fran.sys_admin.lab.% -owner
 % edacl ~con/node_list.dummy -af %.backup.%.% r
 % edacl ~con/node_list.dummy -cf %.%.lab.% -none

    # Check the ACLs.
 % Acl node_list.dummy
Acl for node_list.dummy:
```

```
Subsystem node_list_protect object
 fran.%.lab.%                    pgndwrx
 fran.sys_admin.lab.%            pgndwrx
 %.backup.%.%                    -----r-
 %.%.%.%                         -------

  #Now write the shell script.
  #This is the text of the shell script that Fran wrote.
  #Her assistant uses it to sort node information.

       # Script:  Node_Sort_Program
       # Its pathname is //mc/art/com/node_sort_program.
       #
       # This script sorts nodes in the node list into
       # alphabetical order; then it sorts nodes by node ID.
       # This indicates relative age of the node.
       # Enter the node list name as nodes.^1, where ^1 is the
       # date in yy.mm.dd format.
       #
     subs -up
    srf -s 18 //mc/art/records/nodes.^1 >alphabetical_list.temp
    srf -s 10 //mc/art/records/nodes.^1 >id_list.temp
    catf alphabetical_list.temp id_list.temp >~con/list.^1
           # The next command copies the acl from the dummy program
           # to the sorted file created in this shell script.
    acl //mc/art/con/list.^1 //mc/art/con/node_list.dummy
    args "The file //mc/art/con/list.date contains 2 lists."
    args ´The first list shows the nodes in alphabetical order,´
    args ´The second list shows them by node ID (oldest to newest).´
    dlf alphabetical_list.temp id_list.temp
    subs -down
          # End of shell script

  #Re-enter the subsystem, give the shell script MGR status,
  #and check its status.
 % ensubs node_list_protect
 % subs //mc/art/com/node_sort_program node_list_protect -mgr
 % subs //mc/art/com/node_sort_program
"...node_sort_program" is a NODE_LIST_PROTECT subsystem manager
"...node_sort_program" is a file subsystem data object

  #Leave the subsystem.
 % *** EOF ***
```

Figure 5-1. Example Transcript

Using Sids to Grant Special Access

The registry site directory's **project** file contains the names *sys_admin* and *backup*. Use these names to grant access privileges that will allow certain users to maintain the system.

To enable certain users to perform a system-wide backup, use the **edacct** shell command to create accounts with the project name backup. Then, instruct users to define the following ACL entries for objects that they want backed up:

- Anyone with project name backup must have READ access to the object.
- Anyone with project name backup must have WRITE access to the *backup_history* file in the node's upper-level directory.
- Anyone with project name backup must have ADD access to the directory one or more levels above the object. To enable a user to create or maintain the entire registry database, create accounts with the project name *sys_admin*. Then, grant users with this project name full access (except delete rights) to the registry database.

The system assigns the default project name *server* to any processes created with the display manager command **cps** (Create Process Server). This status is also assigned to DOMAIN system servers described in Chapter 6. Processes with the project name server are independent of log in activity. Use the **cps** command to create processes that run regardless of log in activity, for example, device driver routines. Do not assign the project name server to any accounts.

User.none.none status

The registry site directory files contains an account, ''user.none.none.", which enables anyone to use the network without special rights. You may use ACLs as described in the *DOMAIN System User's Guide* to limit the rights assigned to this SID.

ACLs and Modes

The ACL, or Access Control List, is the DOMAIN system's method of supplying file and directory protection. The following section discusses ACLs in the context of the DOMAIN/IX method of protection, modes. ACLs are actual objects that define who can perform operations on the file or directory, and what operations particular groups of users can perform. ACLs take up space on your disk (at least one block each), so there is incentive to keep the number of ACLs on your system to a minimum.

Normally, the DOMAIN system shares ACLs, by means of the **initial file ACL** and the **directory ACLs**. A newly created file's ACL will be set according to the initial file ACL of the directory in which it is created, and certain forms of access to the new file will be controlled by the directory's other ACLs. The DOMAIN/IX system has an additional mechanism by which it reuses or shares ACL's among DOMAIN/IX files: the **acl_cache**.

Briefly, an ACL entry consists of two parts: the Subject Identifier (SID) and the Access Rights. The SID contains four fields, each of which contains a 64–bit identifier for **person**, **project**, **organization**, and **node**, abbreviated **ppon**. The format of the Access Rights depends on whether the object being protected is a file or directory; these formats are discussed individually below. ACL entries are sorted from most specific to least specific. (If two ACL entries have only one field named and the other fields specified with '%', the leftmost field is sorted first. See the second two samples below.) For example,

```
user.r_d.%.%
```

would be interpreted before

```
user.%.%.%
```

which, in turn, would be interpreted before

```
%.r_d.%.%
```

It is the combination of all ACL entries that makes up the ACL for an object.

We assume here that you understand generally how ACLs are constructed and edited. If not, there is a complete discussion of file and directory ACLs in the *DOMAIN System User's Guide,* Order No.005488. In this book, we only touch briefly on some of the points of that discussion.

The ACL for a file

An AEGIS file has one ACL, that consists of a number of individual ACL entries. The Access Rights scheme for each ACL entry allows or denies the following permissions:

p Protect: change the file's ACLs.

g Grant permission to others.

n Change nodes from which users can access files.

d Delete permission.

r Read permission.

w Write permission.

x Execute object files.

The ACLs for a directory

Each directory on the system has ACL entries, as well. However, a directory is an object that may contain other objects. It may have a different ACL for the directory itself, for files created in the directory, and for subdirectories created under the directory. (Each subdirectory then has three ACLs associated with it, and so on.) The two types of ACL entries which pertain to directories are of the form specified in the next section. The third, which affects files created under the directory, takes the form specified above in the section on file ACLs. For complete details on the interactions possible among these ACLs, see the *DOMAIN System User's Guide*.

The Directory ACL

The directory ACL controls the access to the directory itself. Each entry in the ACL consists of the following Access Rights:

p	Change the directory's ACLs.
g	Grant permission to others.
n	Change nodes from which users can access the directory.
d	Delete the directory.
c	Change names and delete links.
a	Add files and subdirectories.
l	Add links.
r	List entries.
s	Access subdirectories and subdirectory objects.
e	Delete subdirectories and subdirectory objects.

Initial default file ACL

The initial default file ACL is the ACL which is assigned to files created in the directory. Entries in this ACL have a format identical to those in the file ACL described above.

Normally, an AEGIS program that creates a file will use the initial default file ACL. A file created in the directory will **not** share the initial default file ACL if a DOMAIN/IX program created the file, or if the file was created by a program that used the **default_acl**(2) system call. See the *DOMAIN/IX Programmer's Reference* for details of **default_acl**.

At SR9.5, a change to this scheme has been made so that, if the initial default file ACL for a directory is "nil", or unset, then files created in that directory will have "DOMAIN/IX ACLs" applied to them. We provide a new command at SR9.5 to convert the protection scheme of pre–SR9.5 directories from ACLs to DOMAIN/IX modes. See the Administrative Command Reference page in Chapter 11 for **sup**(8), for further information. (See below for a definition and discussion of DOMAIN/IX ACLs.)

Initial default directory ACL

This ACL is the intial default ACL assigned to subdirectories created in the directory. The format of its entries is identical to the format of entries for the directory ACL.

Introduction to DOMAIN/IX modes

The UNIX operating system uses a series of protection bits known as "modes" to limit access to files and directories. In the DOMAIN/IX system, this form of protection coexists with the DOMAIN ACL system. A file or directory's mode is a translation of the ACLs that apply to that file or directory. Obviously, if a file or directory already exists, it has an ACL or ACLs associated with it. If the file or directory was created by a DOMAIN/IX program, subject to the limitations we've mentioned, it will have what we refer to as a "DOMAIN/IX ACL".

Beginning in the next section, we will explain the circumstances under which modes are translated to ACLs, ACLs are translated to modes, and the way in which those things occur. From here on, when there is no need to distinguish in the discussion between a file and a directory, we'll refer to it as an **object**, in order to simplify things.

> Note: The DOMAIN ACL protection system is far more complex than the DOMAIN/IX mode system; you can also set protections for various objects much more specifically with ACLs than with modes. Therefore, if you use the **edacl** command to create a highly specific, multi–layered set of ACLs for an object, do not expect that you will be able to duplicate it with the **chmod** command. This limitation is inherent in the scheme from which the DOMAIN/IX modes are derived, and not a weakness in DOMAIN/IX *per se.*

Translating a DOMAIN/IX mode to an ACL

The "DOMAIN/IX ACL"

When a DOMAIN/IX program creates a file, the DOMAIN/IX mode specified in the program, usually by an **open** or **creat** system call, must be converted to an ACL. There is a specific "DOMAIN/IX ACL" format for this converted mode:

```
person uid.%.%.%    ___________
%.project uid.%.%   ___________
%.backup.%.%        ___________
%.%.%.%             ___________
```

where the line (___________) contains some combination of Access Rights, as previously discussed.

The program first looks in *'node_data/acl_cache* for a "DOMAIN/IX ACL" that corresponds to the mode specified by the **open** or **creat** call. If the program finds a DOMAIN/IX ACL that represents that mode, the ACL is associated with the file that was created, and the reference count of the ACL in *acl_cache* is increased.

If the program does not find an ACL in the *acl_cache* which corresponds to the mode, it must create a new ACL, by translating the mode, **userid**, and **groupid** information into the DOMAIN/IX ACL format. Once that new ACL is created, of course, it is added to the *acl_cache* to keep the number of ACLs on the disk to a minimum.

This is a generalized description of what occurs. Before we get to the specifics of how the actual protection bits of the DOMAIN/IX mode are converted to the Access Rights for the DOMAIN/IX ACL, we need to discuss where the **person uid** and **project uid** parts of the DOMAIN/IX ACL come from.

The user and group IDs

The DOMAIN/IX **userid** and **groupid** of an object define who owns the object, and what group or groups can have access to the object. In AEGIS, the **person uid** roughly corresponds to the DOMAIN/IX **userid**, and the **project uid** corresponds to the **groupid**. The AEGIS **uids**, however, contain 64 bits and the DOMAIN/IX **ids** only contain 32 bits. A file in the node's */etc* directory, called *passwd.map*, functions to translate back and forth between the DOMAIN/IX **ids** and the AEGIS **uids**.

The **person uid** field of the DOMAIN/IX ACL for an object is derived from the **userid** of the object's creator, translated to an AEGIS **person uid** by the */etc/passwd.map* file. In DOMAIN/IX BSD4.2, the **project uid** field of the DOMAIN/IX ACL is the **groupid** of the directory in which the object is created, as translated to a **project uid** by */etc/passwd.map*. In DOMAIN/IX SYS5, the **project uid** field is the **groupid** of the object's creator, again, translated to a **project uid** form by */etc/passwd.map*.

Once the **person uid** and **project uid** fields of the DOMAIN/IX ACL have been determined, the protection bits of the DOMAIN/IX mode must be translated to Access Rights, first for the **person uid** and **project uid**, and then for other users.

Translating the 'pgnd' rights for both directories and files

In creating a DOMAIN/IX ACL entry, the assignment of the **pgnd** rights for files and directories depends on the **userid**. Only the owner of a file, i.e., a user or process with the same **userid** or **person uid** as the file, gets **p** and **n** rights. That is, only the owner can change the object's ACLs or change the nodes from which users can access the object. Since there is no way in DOMAIN/IX to grant a subset of your permissions to another user, the **g** right is always left unset in a DOMAIN/IX ACL. Everyone gets **d**elete rights, except the group 'backup', so the **d** bit is always set. (Note that AEGIS **d**elete rights are not the same as DOMAIN/IX mode delete rights. In order to delete a file created under AEGIS, you must have both delete rights on the file and expunge rights on the directory in which the file resides.) Thus, in a DOMAIN/IX ACL, the first part of the Access Rights for both files and directories will read as follows:

```
    person uid.%.%.%    p_nd . . .
    %.project uid.%.%   ___d . . .
    %.backup.%.%        ____ . . .
    %.%.%.%             ___d . . .
```

Translating Protection Bits to Access Rights

A DOMAIN/IX mode consists of an octal number which specifies permissions for an object. (Symbolic modes do exist; see **chmod**(1).) The octal number is constructed from the logical OR of the following.

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 (in a directory, allow search) by owner
0070	read, write, execute (or search) by group (4, 2, 1)
0007	read, write, execute (or search) by world (4, 2, 1)

For example, an object with the mode **644** would allow the owner to read and write, members of the same group as the owner to read only, and everyone else to read only. (We ignore the first three elements in this list for the purpose of this discussion.) A DOMAIN/IX command like **ls –l** reports an object's octal mode in this form:

```
-rwxrwxrwx
```

where each **rwx** sequence stands for the set of permissions available to user, group, and world, respectively. Our example object with mode **644** above would be reported as:

```
-rw-r--r--
```

Access Rights

The set of Access Rights available for a directory is **pgndcalrse**; each of these rights was described briefly above. The set of Access Rights available for a file is **pgndrwx**. The **pgnd** Access Rights are applied to DOMAIN/IX ACL entries for the appropriate **userid** and **groupid**, as we explained above. The mode of the object created by a DOMAIN/IX program is then translated into the remaining Access Rights, depending on whether the object is a file or directory.

The 'calrse' bits for directories

If the object is a directory, the DOMAIN/IX modes are translated to Access Rights as follows. If the **r** bit is present (for opening or reading the directory itself), it is translated to the AEGIS Access Right **r**. If the DOMAIN/IX mode bit w (for creating or deleting files in the directory) is present, it is mapped to the AEGIS Access Rights **cale**. The DOMAIN/IX x bit (for searching the directory) is translated to the Access Right s.

Therefore, if the directory was created with a DOMAIN/IX mode of **644**, or **rw-r--r--**, the DOMAIN/IX ACL for it would look like this:

```
person uid.%.%.%    p_ndcalr_e
%.project uid.%.%   ___d___r__
%.backup.%.%        _______r__
%.%.%.%             ___d___r__
```

The 'rwx' bits for files

The case for the rest of the Access Rights for a file created by a DOMAIN/IX program is slightly different. The DOMAIN/IX **rwx** bits translate directly to the AEGIS **rwx** Access Rights. If our example file were created with a mode of **644**, or **rw-r--r--**, the DOMAIN/IX ACL would look like this:

```
person uid.%.%.%    p_ndrw_
%.project uid.%.%   ___dr__
%.backup.%.%        ____r__
%.%.%.%             ___dr__
```

There are two exceptions to this direct translation:

The DOMAIN/IX mode -w- becomes **rw-** in Access Rights, and

the DOMAIN/IX mode --x becomes **r-x** in Access Rights.

This is because the DOMAIN/IX system does not support either write-only or execute-only files. In either writing or executing a file under the DOMAIN/IX system, the file must be mapped into the address space of the machine, and read permission must be present for anything to be mapped into the address space.

The fix_cache and flush_cache commands

Once the system has constructed a "DOMAIN/IX ACL" for the object being created, the ACL is added to a node's *'node_data/acl_cache*. This DOMAIN/IX ACL is now available for sharing with other files and directories that may be created on this system by DOMAIN/IX programs.

The *acl_cache* contains DOMAIN/IX ACLs only. It has two major functions. First, and most importantly, it minimizes the number of ACLs on your system, by allowing files and directories created with the same mode to share one DOMAIN/IX ACL. Secondly, it speeds up the translation process from DOMAIN/IX modes to ACLs, and from ACLs to modes. The *acl_cache* has a fixed size, and DOMAIN/IX ACLs can (and will) disappear from it, depending on how recently they were used.

Symptoms of problems with the *acl_cache* are various. We provide two administrative programs associated with *acl_cache* to solve problems: **fix_cache**(8) and **flush_cache**(8). The **fix_cache** program attempts to restore information in the *acl_cache* that may have been corrupted, and the **flush_cache** program clears the *acl_cache* altogether, after which you must reboot the node. These programs are to be used as a last resort only, when there appear to be irreparable problems with the *acl_cache*. DO NOT, for example, run them on a regular basis from */etc/cron*. Complete manual information for these two commands is included in Chapter 11, with the Administrative Commands.

Translating an ACL to a DOMAIN/IX mode

The only time that an ACL must be translated to a DOMAIN/IX mode is when a DOMAIN/IX program (the **ls** command, for example) uses the **stat**(2) or **fstat**(2) system calls, which report file and directory status information, including permissions. Both **stat** and **fstat** will check the *acl_cache* to see if a DOMAIN/IX ACL for the mode of the object exists. If the mode of the object already has a DOMAIN/IX ACL associated with it, the call will then convert the DOMAIN/IX ACL information back to DOMAIN/IX modes, using the same rules that were originally used to translate the mode into a DOMAIN/IX ACL. If the object's DOMAIN/IX ACL isn't in the *acl_cache*, the program will put it there.

If the **stat** system call cannot find a DOMAIN/IX ACL for the object, it must construct a mode from the object's "regular" ACL information. The steps are as follows.

Try to find an owner

First, the program tries to find an owner for the object, that is, an ACL entry that reads

```
person uid.%.%.%    __________
```

In the first entry of this form that it finds, the **person uid** becomes the owner of the file and is translated via the */etc/passwd.map* file to a DOMAIN/IX **userid**. The Access Rights associated with that ACL entry are translated to modes, using the same rules as explained above. If no ACL entry of this form is found, the program searches for an ACL entry of the form

```
person.%.organization.%    __________
```

If the **organization** matches the **organization id** of the program, then those ACL entry rights are translated into the **userid** field of the DOMAIN/IX mode. If neither of these ACL entry forms is found, **userid** is reported as **<none>**.

You may see the **userid** returned by a DOMAIN/IX program (like **ls**) as **<not found>** or **<unknown>**, which is a sign that the */etc/passwd.map* file has been corrupted somehow. In this case, you should rerun the **crpasswd**(8) program.

Try to find group

Next, the program tries to find an ACL entry for the object's **project uid**, in the form:

```
%.project uid.%.%    __________
```

The **project uid** is translated via the */etc/passwd.map* file to a DOMAIN/IX **groupid**, and the Access Rights associated with that ACL entry are translated to modes. Again, it is the first ACL entry of the form that becomes the **groupid**; however, entries of the forms

```
%.backup%.%         __________
%.sys_admin.%.%     __________
```

are skipped, so as to avoid reporting either of these common group names as the **groupid**. If no ACL entry of this form is found, the program searches for an ACL entry of the form

```
%.project uid.organization.%__________
```

If the **organization** matches the **organization id** of the program, then those ACL entry rights are translated into the **groupid** field of the DOMAIN/IX mode. If the attempt to find a **project uid** ACL entry fails, the **groupid** reported by DOMAIN/IX programs for that object will be **<none>**.

Try to find 'world' access rights

Next, the **stat** call will try to find an ACL entry for "world" rights, that is an entry in the object's ACL of the form.

```
%.%.%.%   __________
```

If an ACL entry of this form exists, its Access Rights, too, will be translated to modes. If no ACL entry of this form is found, the program searches for an ACL entry of the form

```
%.%.organization.%        __________
```

If the **organization** matches the **organization id** of the program, then those ACL entry rights are translated into the **world** field of the DOMAIN/IX mode.

A special case

There is a special case where the translation from Access Rights to modes is performed in a non-standard way. If no **person uid** entry was found in the ACL, but there was an ACL entry of the form

```
%.%.%.%
```

or

```
%.%.organization.%
```

where organization matches the **organization** id of the process, then the 'world' **acl** entry rights are copied into the **userid** field of the mode created. If no **project uid** entry was found in the ACL, but the 'world' rights were found, then world rights will be copied into the **groupid** field of the mode created.

Special Note

Under DOMAIN/IX, the first modes that fit are used. That is, if there is a difference between the modes for a given **userid** and a given **groupid**, and the person represented by **userid** is also a member of the group identified by **groupid**, then the mode associated with the **userid** is applied, even if it is more restrictive than the permissions given to **groupid**. An example follows.

Say an object has the following DOMAIN/IX modes.

```
r--rw-rwx           rjc       r_d
```

The **userid** is 'rjc' and the **groupid** is 'r_d'. 'Rjc' has only read rights on the object, and 'r_d' has both read and write rights. Even though user 'rjc' is a member of the group r&d, his or her rights to the object are limited to the first set of modes that fits, i.e., **r--**. This fact is very useful, from the system administrator's point of view, for limiting access to objects within a group.

Overriding DOMAIN/IX protections

In any operating system, there are commands and functions whose use must be restricted. You would not, for example, want all users on your system to be able to run system administration or accounting commands. Access to commands and subsystems of DOMAIN/IX can be restricted by means of modes and DOMAIN/IX ACLs; in order for the system administrator to run these commands, a special **userid** exists, named 'root'. The root **userid** is the most powerful on the system; someone logged in as root can delete any file on the system, for example, including information that the system needs to run correctly. Use of the root **userid** should be severely restricted, preferably to one person on-site at a time.

One of the tasks you have as system administrator is creating the root account and root password, if it doesn't already exist, with the AEGIS **edacct** command; see the installation procedure for details. We strongly suggest you create the 'root' account with a password, and restrict knowledge of the password. It is considered good practice to change the root password periodically.

The 'root' log–in

There are two ways in which you can substitute the 'root' **userid** for your personal userid. The first is to log into the system using 'root' as the log–in name, then supplying the appropriate password.

The second way you may substitute the 'root' **userid**, if you are already logged into the system under another **userid**, is to use the **su**(1) command, which allows you to substitute another userid for yours. See the *DOMAIN/IX Command Reference* for full details about **su**.

Chapter 6

Network and DOMAIN/IX Servers

The first part of this chapter provides general information about servers. It includes discussions about the attributes of servers, starting, stopping and maintaining server processes.

The second part of this chapter provides reference material on each of the servers that arrive with DOMAIN system software. It also includes information on several of the DOMAIN/IX servers (**writed**, **talkd**, and **cron**), including how they are started. The rest of the DOMAIN/IX server and daemon processes are clearly documented in Chapter 7, *Configuring TCP*, so they are not duplicated in this chapter. Server descriptions in this chapter are given in alphabetical order. The reference information contains:

- A description of the server process
- Methods for starting, stopping, and reinitializing the server process
- Information on server configuration files
- Information on server options and arguments
- Examples of options in use
- Special considerations, if any, about the server processes
- References to other information that may be available on the server process.

Two servers, **netmain_srvr** and **ns_helper** have interactive tools. **Edns**, the system administrator's tool for use with **ns_helper**, is described in the *DOMAIN System Command Reference*. **Netmain**, the tool for use with **netmain_srvr** is described in *Managing Your DOMAIN Network.*

Note that this chapter differentiates between creating servers on user nodes (such as the DN4xx/DN6xx or DN3xx) and DOMAIN Server Processors (such as the DSP80), because you use different methods to create servers on DSPs and on user nodes.

General Information on Servers

DOMAIN network server processes:

- Manage user access to network resources, (e.g., printers)
- Manage requests for access to data and the transfer of data
- Gather statistics about the state of the network
- Manage communication pathways outside the network. When properly configured, server processes are transparent to network users, provide cost–effective use of expensive peripheral devices, and enhance the ability of users to share information resources.

Read both the general and particular information on server processes, to decide how to realize the advantages offered by servers in your network.

Methods of Starting Servers

There are several methods used to create servers. The method used affects:

- The attributes of the server
- Whether the server runs in the foreground (with an input window and transcript pad) or in the background (without an input window or transcript pad)
- Whether the server process runs on a local or a remote node.

In most cases, you'll use the Display Manager (DM) process creation commands, **cp** (CREATE_PROCESS), **cpo** (CREATE_PROCESS_ONLY), or **cps** (CREATE_PROCESS_SERVER), to run server processes on a user's node. Issue these commands from the DM input window, or place them in a start–up file. The command and start–up file used dictate the attributes of the server. Commands issued from the DM input window start server processes immediately. Commands placed in start–up files execute when the file executes. Chapter 3 describes both DOMAIN and DOMAIN/IX start–up files and when they execute.

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

Use the shell command **/com/crp** (CREATE_REMOTE_PROCESS) to create processes on any node from a remote node. **Crp** options specify the attributes of the process created. These options provide remote processes with attributes similar to those specified for local processes by **cp**, **cpo**, or **cps**.

Refer to the *DOMAIN System Command Reference* for complete information on the DM commands **cp**, **cpo**, **cps**, and the shell command **crp**.

Use the "**–n** *server_name*" option with server creation commands so that you'll be able to identify a server process in the list output by the **pst** (PROCESS_STATUS) shell command. **Pst** lists the processes running on a node. If you do not give the server a name, and it does not name itself by default, the operating system identifies it as "Process_number." We recommend that you use names for servers (either the defaults that some servers provide, or a name of your choice) to avoid confusion.

Some servers take options and arguments in the DM command line. Generally, when a server processes starts from the DM command line, shell command line features are not used. However, some server programs that start in the DM command line, do use command line features. In these cases, 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.

Server processes can use configuration files for their options and arguments. Some server processes expect configuration files, and these are explained under individual server descriptions. You can create a configuration file for any server that takes options. The file executes when you initiate the server process. The configuration file is called a *names* file. The following syntax:

```
  cps /com/sh -c  server_name configuration_filename
```

causes the **cps** command to create a shell process that takes the server name and configuration filename as arguments. The shell process controls the server, but the server has the attributes of any process started with the DM command **cps**. As with any shell process, a server process created with this syntax can use command line features.

When you use the DM **cp** command to start a server, you can specify the process window size. Precede the DM **cp** command with the coordinates for the upper left and lower right corners of the window, as follows:

```
  (x1,y1)dr;(x2,y2)cp
```

The first set of coordinates is for the upper left corner, the second set is for the lower right corner. See the *DOMAIN System Command Reference* for more information about specifying window sizes.

Attributes of Servers

The table that follows summarizes the information from the *DOMAIN System Command Reference* on the **cp**, **cpo**, and **cps** process creation commands.

Table 6-1. Process Start-Up Attributes

Server Start-Up Method	Does process run in foreground or background?	SID of Process ("id" = node ID)	Process on local or remote node?
cp DM command	Foreground, end on log out.	login account SID	Local
cpo DM command	Background, end on log out.	login account SID	Local
cps DM command	Background, run after log out (except for the siologin server).	user.server.none.id	Local
‘node_data/startup[suffix]	Background, commands with **cpo** or **cps** run whether or not anybody is logged in. **Cp** not used.	user.server.none.id	Local
‘node_data/startup.spm DSP [DSP only]	Background, run whether or not anybody is logged in. **Cp** not used.	user.server.none.id	Local
crp -cpo	Background, end on log out.	login account SID	Remote
crrp -cps	Background, run after log out.	user.server.none.id	Remote

startup_login[suffix]	Commands with **cp** run in foreground, commands with **cpo** run in background. Both end after log out.	login account SID	Local
home_directory/user_data /startup_dm[suffix]	Commands with **cp** run in foreground, commands with **cpo** run in background. Both end after log out.	login account SID	Local
shell command line	Foreground, end after log out.	login account SID	Local
shell command line "&" option	Background, end after log out.	login account SID	Local

Maintaining Existing Servers

Check on the status of network server processes with the shell command **/com/pst**. **Pst** lists all the processes currently 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** output does not show the server process, use the directions in this chapter to restart the server.

If **pst** does not list a server that you started from the DM input window, or by means of a **crp** command or start-up file, the server might not have started properly. ACLs that do not allow access to the server process SID or to the *'node_data* directory can prevent servers from starting. You should not experience this problem if you use the ACLs we provide in the ACL templates. However, if you change ACL entries, and then experience a server start-up problem, check the ACLs for:

- The server program itself. Any person or process starting the program must have EXECUTE rights to the server.

- The *'node_data* directory. Any person or process starting any server must have ADD, READ and WRITE rights to this directory. The SIO line servers have additional requirements. Refer directly to the description of SIO server processes for these requirements.

Pst sometimes lists the server process as running even though the server has stopped. When this happens you must stop the server process explicitly with the **sigp** (SIGNAL_PROCESS) command. Then restart it from the DM window. Refer to the Command Reference for explanations about the **sigp** command. To stop a server process running on a remote node, you must use the **crp** command, log in to the remote node, and then use **sigp** to stop the process.

Alarm Server

The Alarm Server (*/sys/alarm/alarm_server*) alerts you by popping a small alarm window on the screen and sounding an alarm. On-line HELP is available by typing

% /com/help alarm_server

Run the Alarm Server as a background process by starting it with the DM command **cpo**. Do not start the Alarm Server with the **cps** command because the process should not run after a user logs out. Ususally, you'll start the Alarm Server from the *~user_data/startup_dm.[suffix]* file.

The Alarm Server can report on:

- Potential disk overflow. You are notified when the disk containing your entry directory starts to run out of free space.
- Severe network problems. You are notified whenever there is a new network hardware failure message. The hardware failure message is described in the documentation for the **/com/netstat** command.
- Observer reports from **netmain_srvr**. You are notified whenever one of the observers in the **netmain_srvr** program makes a report. This option gives immediate notice when the network has problems detectable only by **netmain_srvr**.
- Brief messages from other users. You can use the **send_alarm** program to direct short messages to other users or nodes. Each condition is checked once every four minutes, or at some other interval set by the **–period** option. Alarms may not be posted every time the condition is checked. See the description of each alarm to find out what that alarm's scheduling policy is.

By default, an alarm is accompanied by a distinctive tone pattern. You may, if you like, 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. Using <CTRL> Q before the message appears kills the Alarm Server.

The Alarm Server names itself "alarm_server" by default, so you need not specify the **–n** option with the **cpo** command.

Certain optional software packages also make use of 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

From the DM command line:

```
cpo /sys/alarm/alarm_server -[options]<RETURN>
```

The server process begins immediately, and ends at log out.

From a *~user_data/startup_dm.[suffix]* file:

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

The server process begins when the named user logs in, and ends when that user logs out.

Configuration Files

To execute Alarm Server options from a configuration file, create a file using the following example format:

Table 6–2. Example Alarm Server Configuration File

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

Edit *~user_data/startup_dm.[suffix]* 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 "*" to point to the executable "options" configuration file. The process names itself "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.

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, 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** alarms 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 detects network hardware problems (as seen in the "last ring hardware failure" report from the **/com/netstat –l** command). The **/com/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 optional *pathname* represents text files containing lists of nodes that run **netmain_srvr.** If no *pathname* is specified, the file *~user_data/alarm_server.netmain_srvr_list* is used. The files should contain lists of node names or hexadecimal node ID numbers, separated by spaces or on different lines. Comments in these files start with { or # characters, and run to the end of the line. More than one *pathname* argument is allowed.

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 in **node_spec** with a hexadecimal ID or node name.

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

-nonetmain (D)

This option prevents the Alarm Server from checking for observer alarms from the **netmain_srvr** program.

-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. This option will not overrride the **-nobell** option.

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 between the horizontal coordinates of each alarm window and **dy** is the difference between the vertical coordinates (in pixels). 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. 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

```
Command:  cpo /sys/alarm/alarm_server -disk 98 -nobell
```

Posts nonaudible alarms when the disk is 98 percent full (2 percent free space).

```
   Command:   cpo /sys/alarm/alarm_server -disk 90 -hw -nm_srvr FFFF5 -w 5 5
```

Posts alarms when: this node's disk is 90 percent full, there is a new hardware failure report message, and whenever there is an observer report from the **netmain_srvr** running on the node with ID FFFF5. Set the alarm window position in the upper left-hand corner of the monitor's screen.

```
   Command: cpo /sys/alarm/alarm_server ´*~user_data/opts´ -n alarms
```

This is a very useful form of the command. It lets you create an options file (in this case *~user_data/opts*) and use that file to control the Alarm Server. The Alarm Server's process is named by the **-n** option. In this case, the process is called "alarms."

Special Considerations

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

The ACLS on two files must allow the **mbx_helper** program read/write access. Make sure that the files

```
`node_data/alarm_server.msg_mbx and
~user_data/alarm_server.msg_mbx
```

both allow read/write access to 'user.server.none'.

Related Information

Information on using Alarm Server with **netmain_srvr** is given in the section on the **netmain_srvr** process.

Additional information is available for using the Alarm Server with certain optional software packages. For example, the DOMAIN Software Engineering Environment (DSEE) is an optional software package that uses the Alarm Server. See the DSEE manuals for additional information, if DSEE is installed on your system. The DOMAIN Professional Support Service (DPSS) also uses the Alarm Server. See the DPSS manuals for additional information if you have DPSS in your network.

mbx_helper – The Mailbox Server (New at SR9.5)

Mbx_helper (*/sys/mbx/mbx_helper*) assists processes on different nodes that communicate using mailboxes. **Mbx_helper** must run on any node that sends or receives mailbox communications across the network. Beginning at SR9.5, the **mbx_helper** process is automatically started by any Apollo server or program that needs it to operate correctly. For example, to send a message via the **/com/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 startup script, if you are running a non-Apollo server that requires **mbx_helper** to operate.

On-line HELP is available by typing

% /com/help mbx_helper

For further information about starting **mbx_helper** from programs, see *Programming With System Calls For Interprocess Communication*. For purposes of providing network services to a node, enable **mbx_helper** by removing the # sign from the appropriate *'node_data/startup[suffix]* script on that node.

Special Considerations

In a secure network, a mailbox receives its ACL 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 ACLs you do 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 System Calls For Interprocess Communication* for more information. Note that you cannot change a mailbox's ACL while the mailbox is in use.

netmain_srvr – The Network Maintenance Server

The **netmain_srvr** (*/sys/net/netmain_srvr*) collects data necessary for maintaining the network. Use the data for monitoring network performance and isolating potential problems. Use the Netmain Interactive Tool, described in *Managing Your DOMAIN Network*, 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 network. 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 certain unusual conditions. If an observer detects such a condition, it can provide an alarm on the node running a monitor.

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. **Netmain_srvr** log files reside in *'node_data/net_log* directory by default.

If **netmain_srvr** does not start properly, a record of the failure appears in *'node_data/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. **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. **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 *DOMAIN System Command Reference* provides information on **netmain_chklog** and **netmain_note**.

Netmain_srvr should run as a background process from *'node_data/startup.[suffix]* 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 DSPs. Whenever possible, run a monitor in each loop so that data can be collected even if the loop is switched out of the main network. 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.

On-line HELP is available by typing:

```
% /com/help netmain
```

for instructions on controlling **netmain_srvr** after it starts, and on analyzing the data collected. Type

```
% /com/help netmain_note
```

for information about adding notes to the network error log. Type

```
% /com/help netmain_chklog
```

for information about detecting and deleting corrupt log files.

Data Collected by netmain_srvr Probes and Observers

This section 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

Records performance statistics about each node's CPU usage, and writes them to: *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

Aegis CPU time: Shows the time the operating system on each node spends on internal needs, and on servicing low-level 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.

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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

DISK_ERRS - Disk and storage module errors

Records cumulative information about disk and storage module performance and errors on all nodes and writes them to *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

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) errrors 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%, for example, reads account for 49% 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%, for example, writes account for 51% 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.

Over–runs: 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%, for example, reads account for 51% 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%, for example, writes account for 51% 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 IO 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 IO rates per unit time.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

ERR_COUNTS – Network error counts (normal traffic)

Records cumulative network performance statistics and error counts on all nodes and writes them to *'node_data/net_log/net_log.yy.mm.dd* or a file you specify.

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 perfor mance 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 **/com/netstat** 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.

Over run, 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.

Over run, 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.

Time out, receive: Counts messages received that did not complete in the exepected time. The display formats show the statistic as a percentage of the node's total network receives.

Time out, 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% 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%, for example, incoming messages account for 43% 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%, for example, outgoing messages account for 43% 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

EST_TOPOLOGY – 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.

Records every change in the hardware failure message reported by the **/com/netstat** command, on the node that is running the monitor. The ring hardware failure 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.

Hardware failure messages are generated by AEGIS. They appear when there is no network traffic, and AEGIS is completely unable to send network messages.

Default Probe Interval Time: 0:01:00

Default Probe Skip Distance: Not Applicable

MEMORY – Records counts of memory errors on nodes in the network.

Memory errors shown in printed output formats, list 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.

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.

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 zero 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. When 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 zero 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 look-ups).

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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

PAGING – Diskless/partner information.

Records information about diskless nodes and their paging partners.

Diskless nodes/paging partners:
This formatting option produces a display of diskless nodes and their paging partners. This display can not be used with data taken from a running monitor. Use it to analyze data from log files.

Paging partners, ordered by diskless node:
This option shows the node(s) used as paging partner(s) by each diskless node.

Paging partners, ordered by mother node:
This option shows the diskless node(s) supported by each node that that volunteered as a paging partner.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD_10_MSGS – Software Diagnostic Messages (10).

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.

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 software diagnostic 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 over run: 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 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 time 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.

Default Probe Interval Time: 0:30:00

Default Probe Skip Distance: 1

SWD_100_MSGS - Software Diagnostic Messages (100).

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

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. For more information on probe offset times, see *Managing Your DOMAIN Network.*

Default Probe Interval Time: 3:00:00

Default Probe Skip Distance: 1

TOPOLOGY - Total node list estimate.

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.

Default Probe Interval Time: 1:00:00

Default Probe Skip Distance: Not Applicable

MODEM_ERRS

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

This observer reports on nodes that have more than 0.01% of Winchester disk drive CRC errors.

Default Probe Interval Time: 0:30:00

Default Recheck Interval: 12:00:00

Starting and Stopping netmain_srvr

From the DM command line:

```
cps /sys/net/netmain_srvr options_filename [-options]<RETURN>
```

The process begins immediately and continues after log out.

From the *'node_data/startup.[suffix]* file:

```
cps /sys/net/netmain_srvr  configuration_filename [-options]
```

The process begins when the node comes on line and continues after log out. 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 the shell command **sigp** as shown:

```
% sigp netmain_srvr -q
```

The process stops running if the node is shut down intentionally or because the system crashes. If the process stops running you may start it from the DM command line or by re-booting the node.

Options and Arguments

Netmain_srvr has a number of options. Instead of including them all on the command line, you can use an options file, by specifying the **-cmdf** option. If you specify **-cmdf** *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.

OPTIONS

Default options are indicated by (D).

-append Appends to an existing log file the name specified by the **-l** option, 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]* 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, 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 filename 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* (D)

Sets 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 if you don't want to use the default length for the first log file, since you cannot change the length of a log file once it's open.

-nappend(D)

Creates a new log file, overwriting 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

Instructs **netmain_srvr** to run probes and observers, but not to write data to a log file. A monitor without a log file can broadcast status messages to other monitors which can store the information in their log files. Choose **-nl** when you run a monitor on a node with limited disk spaced. You can open log files when disk space is available with the Netmain Interactive Tool.

-ntopo[_init] (D)

Overrides the **-topo_init** option, if **-topo_init** is specified in an options file. The **-ntopo** option is useful only on the command line.

-obs[erve] *observer time ...*

Sets 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.

-re_obs[erve] *observer time ...*

Sets 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 insure 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.

-s[ample] *probe time ...*

Sets 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.

-sk[ip] *probe distance ...*

Sets 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.

-topo[_init] *pathname*

Initializes 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, (i.e., # and { are comment markers).

Examples

```
Command: cps /sys/net/netmain_srvr -ll 1500 -l tuesday_error_log

Command: cps /sys/net/netmain_srvr -s error_counts 0:15 hw_fail never

Command: cps /sys/net/netmain_srvr -cmdf /sys/net/start.net_srvr -ll 3000
```

Table 6-3 contains an example configuration file for **netmain_srvr**.

Table 6-3. Example netmain_srvr Configuration File

```
# The file /sys/net/start.net_srvr might contain
# these lines:
-l -ll 1500
-sample err_counts 0:01:00 -skip err_counts 30
-sample topology 0:20:00
-sample disk_errs 0:01:00 -skip disk_errs 30
-sample time_skew never
-observe modem_errs 0:10:00
```

Special Considerations

To read the data in the *‘node_data/netmain_srvr.err.log* error log file, copy the file to a second file and then read the data in the second file. The error log file is locked while you read it so the monitor will not be able to write any error messages to it. Copying the file to another file locks it very briefly.

Following is an example of error messages in a file copied from an error log file.

Table 6-4 Example netmain_srvr Error Log

```
22:21 1984/3/14 -- ?(netmain_srvr) object is in use (OS/file server)
11:18 1984/3/20 -- ?(netmain_srvr) -log and -nlog must not be used together.
11:23 1984/3/20 -- ?(netmain_srvr) One or more unrecognized flags
                   in the command line.
```

If you cannot detect **netmain_srvr** problems by reading the error log, use the DM **cp** (CREATE_PROCESS) command to run **netmain_srvr** in a window. You may see other error messages that can explain the problem. Once you have eliminated the problem, rerun **netmain_srvr** using **cps**.

Using the Alarm Server with netmain_srvr Observers

Observers collect data each time a probe is activated. Observers can alert you to potential problems in your nodes or network. Run the Alarm Server with the **netmain_srvr** options, to receive notice about observer reports. When it detects problem conditions, the observer signals the Alarm Server to open a small alarm window on the node running the monitor, and to record some more information in a file.

To receive notifications from observers, start the Alarm Server with the **-netmain** *[pathname]* or **-nm_srvr** *node_spec* options. To receive notification from **netmain_srvr** observers running on multiple nodes, use **-netmain** *[pathname]* in the alarm server start-up command.

If specified, *"pathname"* is a text file containing lists of node names or hexadecimal node IDs. You may specify multiple pathnames as arguments. If you do not specify a pathname, you must give the text file the default name: *home_directory/user_data/alarm_server.netmain_srvr_list*. If you want to use both the default file name and additional text files, you may specify **-netmain** several times in the alarm server start-up command, for example:

```
cpo /sys/alarm/alarm_server -netmain pathname pathname -netmain
```

In the text files, the lists of node names or hexadecimal node ID should be separated by spaces or on different lines. Certain combinations of characters could be either node names or hexadecimal IDs, for example: feed3. If the text files contain such combinations, indicate a node name with // (e.g., //feed3), or indicate a node ID by preceding it with zero, i.e., 0FEED3. To include comments in these text files, prefix a comment with a "{" or "#" character. The comments can start anywhere in the line and must run to the end of the line.

The alarm server reads node list files only when it starts up. If you add or delete node names after the server is running, the changes do not take effect until the next server startup.

Any node can run the Alarm Server and receive observer alarms. It does not itself have to be a node running **netmain_srvr**.

The Alarm Server reports the name of the observer, the node, and the time at which the problem occurred. The observer keeps a more detailed list of observer reports in the file *'node_data/net_log/observer_name.obs*. Substitute *win_crc* or *modem_errs* for *observer_name* in the log pathname to see a long observer report.

The example below shows part of an observer report list file. The node(s) on which the problems were reported appear at the end of each report.

```
Noted at 1984/03/21 11:31:24 by observer win_crc on the_hedge's server.
 Excess disk CRC errors seen on nodes:
 fossil   1.00 / 76.00 (1.3%)

Noted at 1984/03/21 14:13:57 by observer win_crc on the_hedge's server.
 Excess disk CRC errors seen on nodes:
 mantis   61.45 / 1618.20 (3.7%)

Noted at 1984/03/22 13:22:13 by observer win_crc on the_hedge's server.
 Excess disk CRC errors seen on nodes:
 perseus  424.61 / 1833.70 (23.1%)
```

netman – The Diskless Node Server

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

The **netman** process, running on a disked node, receives "request for volunteer" broadcasts from diskless nodes attempting to boot. **Netman** 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 node's 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 Chapter 3 for more information on node start-up directories. In addition to the **netman** process, there exists a DOMAIN/IX shell script, named **mkptnr**, which creates the files necessary for running DOMAIN/IX on a diskless node. **Mkptnr** places these files in the node's /sys/*node_data.diskless.node_id* directory. Complete information, including what files it creates, and how to invoke the **mkptnr** script, are included in the Administrative Command reference page for **mkptnr**(8) in Chapter 11 of this manual.

The netman server runs as a background process from the *'node_data/startup.[suffix]* file. The command line that executes **netman** is in the default */sys/dm/startup_template* script that arrives with your system. Remove the "#" sign at the beginning of the command line to enable the process. **Netman** will execute from the start-up script when the node is re-booted. You may start the process from the DM command line as shown below.

On-line HELP is available by typing

```
% /com/help diskless
```

Starting and Stopping netman

From the DM command line:

```
cps /sys/net/netman<RETURN>
```

The server process begins immediately and persists after log out.

From the *'node_data/startup.[suffix]* file:

```
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 from the DM command line.

Special Considerations

Netman allows any disked node to continue functioning, even if its disk becomes non-functional. For this temporary procedure, get the node ID of any node running **netman**. (You may wish to start the **netman** process on a disked node in the same loop as the now diskless node.) Use the DM command **shut** to bring the node that you want to boot diskless down to the Mnemonic Debugger. Then type:

```
> RE<RETURN>
> DI N NODE_ID<RETURN> {node_id = node running netman}
> EX AEGIS<RETURN>  Network Partner ID nnnnn
```

Note that in this procedure **netman** bypasses the */sys/net/diskless_list*. It isn't necessary to edit the list to get the node back in the network. When you use this procedure **netman** will create a *node_data/ startup.[suffix]* file for the node with disk problems.

New at SR9.5 - init_tmp_dirs

Formerly, a program to initalize the temporary directories */tmp* and */usr/tmp* was included in the system software directory */sys/dm* as the executable file *init_tmp_dirs*. The function of */sys/dm/init_tmp_dirs* is now provided by **netman**, and */sys/dm/init_tmp_dirs* has been eliminated. See the DOMAIN/IX Release Notes for SR9.5 for further information.

ns_helper - The Naming Server Helper

Ns_helper (*/sys/ns/ns_helper*) manages a master root directory. This database is the only comprehensive source of node identifying information in the network. A later section in this chapter describes the procedures you follow to implement **ns_helper** in your network.

Ns_helper performs most of its operations automatically. **Edns**, an interactive tool used with **ns_helper** is available for those operations requiring your intervention, for example updating the database.

Ns_helper maintains a cache of the master network root directory at each node. Whenever the naming server uses the 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 -up-**

date" 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. **Ns_helper** propogates changes in the database of any replica to all other replicas for a period of fourteen days. In exceptional circumstances of node, loop, or disk failure, a replica may not receive updated information in this time period. Use **edns** ''merge" 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 *'node_data/startup.[suffix]* file so that it will continue after log out. **Ns_helper** names itself ''ns_helper" by default, so you need not specify the **-n** option to the process creation command.

Starting and Stopping ns_helper

From the DM command line:

```
cps /sys/ns/ns_helper<RETURN>
```

The server process begins immediately and persists after log out.

From the *'node_data/startup.[suffix]* file:

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

The server process begins when the node comes on line and continues after log out.

With both start-up methods described above, **ns_helper** will continue running until it is intentionally stopped with the shell command **sigp**, as shown:

```
% sigp ns_helper -q
```

If **ns_helper** stops running, restart it from the DM command line, or, if necessary, by rebooting the node.

Special Considerations

When you run more than one **ns_helper** process in your network, place a server inside and outside of loops which are switched out regularly.

prsvr – The Print Server

Prsvr (*/com/com/prsvr*), the print server process, manages files submitted to the print queue with the shell command **prf** (PRINT_FILE) or the menu driven print command, **prfd** (PRINT_FILE_DISPLAY). Create a print server by executing the **prsvr** command and specifying the configuration file for the printer type. This command should execute only when you start or restart the node connected to the printer. Do not execute the command at other nodes: this will cause print files to be lost.

The print server supports the following printers:

1) NEC Spinwriter 7715, connected via an SIO line.

2) Printronix 300/600 LPM printer, connected to the PBU.

3) Versatec V–80 or V–82 series printer–plotter, connected to the PBU via an IKON 10071 or 10085.

4) GE series 3000 printer, connected via an SIO line.

5) IMAGEN LBP10 and IMAGEN CX printers, connected via one of the following: SIO line; Versatec IKON 10071 or 10085, or user–supplied Multibus controller.

6) DOMAIN/LASER–26, a 300 dots–per–inch, 26 pages–per–minute laser printer, connected via an SIO line or IKON 10085 multibus controller.

7) APPLE LaserWriter, a 300 dots–per–inch, 8 pages–per–minute laser printer, connected via an SIO line.

Typically, print servers are started as background processes from the *'node_data/startup[suffix]* file. This is the recommended start–up method for printers that are available to the entire network at all times. There are other start–up methods you can use. Select these methods on the basis of the node that hosts the print server, and the attributes you want the **prsvr** process to have.

On–line HELP is available by typing

```
% /com/help prsrvr
```

or

```
% /com/help prsrvr config
```

for details about device configuration files. Type

```
% /com/help prsrvr device_drivers
```

for details about writing user–supplied device drivers.

Starting prsvr

From the DM command line, on the local node:

```
cps /com/sh -c ´/com/prsvr´ configuration_filename -n process_name
```

The server process begins immediately and continues after log out.

From the shell command line, to start a prsvr process at the time you want to use the printer:

```
% prsvr [configuration_filename] [&] [options]<RETURN>
```

Configuration file name (optional)
You may specify a name or the default name is *printer_config.data*.

& (optional) This character creates a separate shell process in which to run the print server. This process is created without the normal pads or windows, thus running invisibly in the background. To stop this background process before log out, use the **sigp** command.

–n (optional) If you do not use the "**–n**" option, the print server process is called *print_server.printername* by default.

At NODE startup:

The *'node_data/startup[suffix]* or *'node_data/startup.spm* file starts the process if you remove the comment character (#) from the following line in those files:

```
# cps /com/sh -n print_server
```

Edit this line to include the name of the print server configuration file. Place the configuration file in any directory; however, it usually is convenient to place it in */sys/print*. For example,

```
cps /com/sh  /sys/print/spinwriter -n print_server
```

Unless directed elsewhere, **prsvr**'s default search for the configuration file starts in the current working directory. **Prsvr** follows the operating system's usual search rules (defined in the *DOMAIN System User's Guide*.)

The server process begins when the node comes on–line and continues after log out.

From a REMOTE NODE:

From a remote node, the shell command **crp** starts the print server on nodes running **spm**. Use the **crp** command in one of three ways:

The following method:

```
crp /com/prsvr config_file -n print_server -on node
```

creates a window to the process on the node and prompts you to log in. Processes started with this invocation of **crp** run in the foreground and end at log out. That is, they have the same attributes as a process started with the **cp** command from the DM input window (see Table 6–1).

The next method:

```
crp /com/prsvr config_file -n print_server -on node -cpo
```

runs the process in the background (there is no window to the process), and the process ends at log out. These are the attributes of processes started using the **cpo** command from the DM input window.

A final method:

```
crp /com/prsvr config_file -n print_server -on node -cps
```

starts the process in the background and the process continues after log out. These are the attributes of processes started using the **cps** command in the DM input window (see Table 6–1). A process started with this method can be stopped with the shell command **sigp**. It must be restarted if the node stops running the process for any reason.

Stopping prsvr

With the DM command and the *'node_data* start–up methods described above, **prsvr** will continue running until it is intentionally stopped with the shell command sigp, as shown:

```
% sigp prsvr -q
```

With both start-up methods described above, the process stops running if the node is shut down intentionally or because the system crashes. If it stops running, restart the process from the DM command line, or by re-booting the node.

There is a special application for

```
% sigp prsvr -s
```

which tells **prsvr** to, "stop printing a file that is currently printing, and delete the filename from the print queue." If a file is not currently printing, the **-s** option reverts to its usual meaning, i.e. "stop the process entirely."

> NOTE: **sigp -s** should be used in this manner only with line printers, for example a Spinwriter. Do not use it with printers that have intelligent print controllers, such as laser printers, or machines where you can delete waiting jobs.

Configuration Files

To start printers other than the default printer (p), or to use **prsvr** options, create a printer configuration file. Table 6-5 shows example configuration files.

Table 6-5. Example Print Configuration Files

```
PRINTRONIX PRINTER                               IMAGEN CX PRINTER

PRINT_WIDTH     13.0                             PRINT_WIDTH      8.0
PRINT_LENGTH    11.0                             PRINT_LENGTH    10.8
BOTTOM_MARGIN   4                                PAGE_HEADERS    OFF PAGENO_COL-
UMN  90                                          PAGENO_COLUMN   72
RESOLUTION      66                               FILE_BANNERS    ON
FORM_FEEDS      1                                DEVICE          IMAGEN
FILE_BANNERS    ON                               PLOT_MODE       ON
PAGE_HEADERS    OFF                              INTERFACE       SERIAL
PLOT_MODE       ON                               SPEED           19200
PRINTER_NAME    P                                PRINTER_NAME    CX
DEVICE          PRINTRONIX                       SIO_LINE        1
LOGO            <NONE>                           RESOLUTION      300
                                                 LOGO<NONE>
```

You can specify the name of the configuration file in the start-up file, for example:

```
cps /com/sh `/sys/print/ptx_config` -n print_server
```

If you use the shell command **crp** to start the print server, type:

```
% crp ´/com/prsvr /sys/print/ptx_config´ -n ps -on node -cps<RETURN>
```

In each example above, the configuration file is named "*ptx_config*" (for the Printronix printer). Give a different name to each printer configuration file when there are several printers attached to a node. If you do not give the print server configuration file a name, the default name is *printer_config.data* (located in the current working directory).

Print Configuration File Options and Arguments

Some of the options and arguments below can now be overridden by user options to the **prf** command. Additionally, some new options make older options obsolete. The older options still are functional, and are the defaults in some circumstances. See individual descriptions for complete explanations.

BOTTOM_MARGIN [n]
Number of lines to skip at the bottom of the page. The **prsvr** uses this option only if the **prf** or **prfd** command does not contain a margin specification. Default value is 4 lines.

COLLATE_COPIES[ON|OFF]
Enables page collation for multiple copies if ON.

CPI [n]
The default characters per inch (the pitch) printed by the current printer. May be overridden by user options to the **prf** command. Use this option if you use a non-default pitch for a printer. The following list shows the defaults for each of the printers we support:

IMAGEN	12 cpi
SPIN	12 cpi
PRINTR	10 cpi
VERS	12 cpi
GE	12 cpi

See also PRINT_WIDTH.

DDF_NAME *[pathname]*
Specifies a device descriptor file for a MULTIBUS controller. This option is useful for driving a printer with Multibus card. Apollo supplied printers that may use this option are the Imagen and Versatec printers. Default name is */dev/versatec*. See the *GPIO User's Guide* for more information on this option.

DEVICE [SPINWRITER | PRINTRONIX | VERSATEC | GE | IMAGEN | USER1 | USER2 | USER3 | USER4]
Specifies the DOMAIN supported printer types or user supplied device drivers which print files. See reference material on the USERn argument. Default device is the Spinwriter.

FILE_BANNERS [OFF|FIRST|LAST]
Causes a banner page to precede each file. May be overridden by user options to the **prf** command. OFF means no banner page; FIRST means the banner page is printed before the job and LAST, after the job.

FORM_FEEDS [n] The number of pages to form feed between jobs. Default is 1 form feed (paper advances from current page to top of next page.).

INTERFACE [SERIAL | VERSATEC | MULTIBUS]
Indicates the hardware interface used by an IMAGEN printer. The hardware interfaces are: SERIAL – SIO line; VERSATEC – Versatec IKON 10071 and 10085 boards; MULTIBUS – user supplied MULTIBUS controller.

LPI [n]
The number of lines per inch printed by the current printer. Use this option if you don't use the default number of lines per inch on a printer. The following table lists the defaults for each of the printers we support:

IMAGEN	6 lpi
SPIN	6 lpi
PRINTR	6 lpi
VERS	6 lpi
GE	6 lpi

See also PRINT_LENGTH.

LOGO [string|none] Specifies a character string to be printed on the banner page. Default is none (no logo is printed)

MOdel_NUMBER [xxx]
Currently, this option applies to printers with multiple model numbers, i.e.:

IMAGEN	8/300	for the CX
LBP-10		for the LBP-10
VERSATEC	V8236	
	V8224	
	V8244	
	V8272	
	V80	

PAGENO_COLUMN [n]
Column in which page number is printed in the header. May be overridden by user options to the **prf** command. Default column is 90.

PAGE_HEADERS [ON|OFF]
ON prints a one line header at the top of each page, containing the filename and page number. May be overridden by user options to the **prf** command. Default is ON.

PAGE_LENGTH [n]
Length, in lines, of page. This is the default option if PRINT_LENGTH is not specified. Default length is 66 lines.

PAGE_REVERSAL[ON|OFF]
Order of pages printed. If ON, the pages of the file are printed last page first; if OFF, first page is printed first. (Supports PostScript interpreter.)

PAGE_WIDTH [n] Width, in characters, of page. If the input line length exceeds the specified page width, the excess charaters are truncated and a warning message appears, listing the number of truncated lines. This is the default option if PRINT_WIDTH is not specified. Default width is 132 spaces

PLOT_MODE [ON|OFF]
Specifies whether the device will accept plot files. The default mode is OFF.

PRINT_LENGTH [inches]
Specifies in inches the length of paper that can be used for printing, i.e., the length of paper minus any limits set by the printer's physical capabilities, including the margins you physically set on the printer. For example, Imagen printers will take paper size 8.5 x 11 inches but can only print on an area size 8.00 by 10.8 inches. Use this command instead of PAGE_LENGTH to specify page format. See also the example configuration files and the LPI option.

PRINT_WIDTH [inches]
Specify in inches the width of paper that can be used for printing, i.e., the width of paper minus any limits set by the printer's physical capabilities, including the margins you physically set on the printer. For example, if you use Spinwriter operator settings, the PRINT_WIDTH option must reflect those settings. Use this command instead of PAGE_WIDTH to specify page layout. See also the CPI option.

PRINTER_NAME [string]
Specifies printer name used in the "**-pr**" option of the "**prf**" command; useful when several printers are attached to a single node. The default string is p.

RESOLUTION [n] Specifies the resolution of the printer in dots per inch. If you use the IMAGEN CX, set the resolution at 300 dots per inch. You may also choose a resolution of

144 dots per inch for the GE printer if the firmware configuration is 403277 or greater. May be overridden by user options to the **prf** command. Defaults are: CX: 300; LBP10: 240; and GE: 72 dots per inch.

SIO_LINE [n] Specifies SIO line to which a printer is attached; not meaningful for Printronix or Versatec printers. Default is line 1.

SPEED [n] Specifies baud rate for the SIO line; not meaningful for Printronix or Versatec printers.

TOP_MARGIN [n] Number of lines to skip at the top of the page. **Prsvr** uses this option only if the **prf** or **prfd** command does not contain a margin specification.

The DEVICE USERn Option - User Written Device Drivers

Prsvr can support as many as four customer-supplied device drivers on systems running Software Version SR4.1 or later. This support is in addition to that supplied for the NEC Spinwriter, Printronix, Versatec, Imagen, and GE 3000 printers.

If you write your own device driver to run a printer you supply, it must include the six procedures listed below. Use the names shown. Replace 'n' with the device number that corresponds to the number in the DEVICE USERn option in the configuration file.

The declarations of the following calls and data types used may be found in */sys/ins/prsvr.ins.pas.*

- USERn_INIT (sio_line, sio_speed) -- **prsvr** calls this procedure once to initialize the printer. The procedure typically initializes the internal state of the driver. If you are using an SIO line, SIO_$CONTROL should be called to setup the characteristics of the line. The SIO_SPEED parameter, declared in */sys/ins/sio.ins.pas* or */sys/ins/sio.ins.ftn*, indicates the line speed of the selected SIO line (i.e., one of SIO_$50, SIO_$75, SIO_$110, . . . SIO_$19200). A stream call should be made to open the SIO line.

- USERn_WRITE (string, length) -- **prsvr** calls this procedure to pass the specified string to the device. The string may be ASCII text and include zero or more newline characters, or it may be plot data. It is the driver's function to correctly interpret this data.

- USERn_SET_MODE (mode, value) -- **prsvr** calls this procedure to pass the device driver a pointer to an attribute block. The attribute block contains information that the user may have specified at the time the file was enqueued. Refer to */sys/ins/prsvr.ins.pas* for the contents of the attribute block named "SERVER_DB_T."

- USERn_RETURN_INFO (query, info) -- **prsvr** issues this call to determine the capabilities of the device from the user-supplied device driver. The data is passed to the server as a pointer to a record structure called "DRIVER_DB_T", defined in */sys/ins/prsvr.ins.pas.*

- USERn_FLUSH -- **prsvr** calls this procedure to flush the output buffer for the device. Ignore this call if the device driver does not buffer data for the device.

- USERn_CLOSE -- **prsvr** calls this procedure to close the device.

After you write the device driver, bind it to the print server as follows:

```
% /com/bind -b prsvr.user /com/prsvr usern.bin<RETURN>
```

The binder will notify you of undefined globals. For device driver *usern.bin,* ignore all undefined globals except those that refer to *usern.*

The INTERFACE MULTIBUS Option - Interfacing the Imagen Printer

The INTERFACE MULTIBUS option for the laser printer allows you to use MULTIBUS controllers *other than* the Versatec IKON 10071 and 10085 boards. To use this option, write a program module containing the following entry points:

```
multibus_$init(sio_line,sio_speed) performs any required initialization.

multibus_$write(buf,buflen) passes a buffer (buf) of type univ pr_$buf_t and
length (buflen) to the controller.
```

These entry points should be specified in the DDF for the controller. The actual binding takes place at run time. Be sure to specify the DDF_NAME in the **prsvr** configuration file DDF_NAME option.

Special Considerations

Note how the default printer_name (p) is used in the example Printronix configuration file. Users can send files to this printer, without using the "**–pr**" option in the shell command "**prf**". In the example IMAGEN file, the **printer_name** option is specified as "cx." Users must specify "**–pr cx**" when they send files to this printer.

You may configure printers so that, if one is busy, another one will automatically start printing. You do this by specifying configuration files with the same PRINTER_NAME option.

You must create a link, on every node in the network, to the */sys/print* directory of nodes connected to printers. You can distribute printer resources by linking some users to one */sys/print* directory and linking other users to a different */sys/print*. Users always can specify different */sys/print* directories with the **–s** option of the **prf** command.

The new print configuration file options, CPI, LPI, PRINT_LENGTH, and PRINT_WIDTH are meant to be used together to give prsvr information on page format. Note that CPI and LPI would be used only if you are using non-default options on your printer. Usually PRINT_LENGTH and PRINT_WIDTH will be sufficient to set up a page format. Use these options in preference to the options PAGE_LENGTH and PAGE_WIDTH.

Related Information

Refer to the documentation for the GPIO package for further details on using the MULTIBUS and writing device drivers.

SIO – Serial I/O Line Servers

The SIO line servers allow you to connect a "dumb" terminal to a DOMAIN workstation, either directly, or via modem and telephone lines from another location. These servers manage a DOMAIN workstation's asynchronous I/O (SIO) lines, and the process of logging on to the connected terminal.

The SIO line server processes are:

siomonit [SIO_PROCESS_MONITOR] (*/sys/siologin*),
typically invoked as a background server process from the *'node_data/startup[suffix]* file.

siologin [SIO_LINE_LOGIN] (*/sys/siologin*),
invoked by **siomonit.** It must be a manager within the ''login'' protected subsystem. **Siologin** is the process that directly manages user log in.

Below, we describe the procedure used to connect terminals to DOMAIN 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:

- Use the shell command **/com/tctl** to display the SIO line configuration. Default values for the SIO lines are:

  9600 baud
  No parity
  Eight bits per character
  One stop bit

- 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.

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

- 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 EOT 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.

siologin – The SIO Line Login Server

FORMAT

siologin dev_name [[-dialin] [-n name] prog [args...]]

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 login 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*). "**Dev_name**," which must be specified, is the SIO device descriptor pathname. Other options, if specified, must precede "**prog**" and its arguments. The **siologin** sub-system 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 will look for a start-up file *‘node_data/startup_sio.sh* and if it exists, will execute it as a shell command file, passing it the SIO line number as an argument, e.g. for */dev/sio1*:

```
/com/sh ´node_data/startup_sio.sh 1
```

Include **tctl** commands here to configure the line or execute

```
% ulkob ^1 -f<RETURN>
```

to ensure that the SIO line is not locked through some previous failure.

On-line help is available by typing

% /com/help siologin

or

% /com/help protection protected_subsystems

siologin Options and Arguments

Arguments:

dev_name (required)
The SIO device descriptor pathname, in the form */dev/siox*, 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 log in is complete. If omitted the default invokes */com/sh* to start the shell command line interpreter.

args
Arguments for the program specified in **prog**.

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 log in sequence.) **Siologin** logs invalid login attempts in *‘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 login 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. Table 6-6 shows an example log file.

Table 6-6. Example *'node_data/siologin_log*

```
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.
```

Note that to operate, the **siologin** server must have manager status in the "login" protected subsystem (see the *DOMAIN System User's Guide* for more information on the subsystem.) The DOMAIN 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<RETURN>
```

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:

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

Siologin has lost its manager status. To reassign manager status to **siologin**, you must log in with a *sys_admin* account, and type:

```
% ensubs login<RETURN>
% subs /sys/siologin/siologin login -mgr<RETURN>
% <CTRL> Z
```

siomonit – The SIO Process Monitor

Siomonit supports repeated logins over SIO lines, independent of any log in/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 above. **Siomonit** 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 # sign.

Special Note for DOMAIN/IX Users

In order for a user who is logged in over an SIO line to be able to use DOMAIN/IX environment variables (including SYSTYPE), the lines in the node's startup files that start **siologin** must appear after the lines which set the DOMAIN/IX environment variables. That is, **siomonit** and **siologin** should be started after the DOMAIN/IX environmental variables are set in the node startup files.

Starting siomonit

From the DM command line:

```
          cps /sys/siologin/siomonit siomonit_filename<RETURN>
```

The server process begins immediately and continues after log out.

We recommend this start-up method if you use **siomonit/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*.

From the *'node_data/startup.[suffix]* file,

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

Be sure this line appears after any lines in the startup file which set environment variables, including the uncommented line of the form

```
  env SYSTYPE ´bsd4.2´
```

The server process begins when the node comes on line and continues across log ins and log outs at the node terminal. We recommend this start-up method if you use **siomonit/siologin** all the time.

Signaling the siomonit Process

Sometimes you will wish 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 <RETURN>
```

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* (see below) file first, to determine why the process stopped.

Example Siomonit_file

Siomonit_file name argument lists have the form:

[-repeat] siologin_arg_list

Contents of the siomonit_file

-repeat

configures **siomonit** to restart this **siologin** process after a user logs out, i.e., 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 above. Arguments are passed to **siologin** unvalidated, however the first argument must be */dev/sion*. **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.

Table 6-7 shows an example *'node_data/siomonit_file*. Note that comments may be included by placing the "#" sign at the beginning of a line.

Table 6-7. Example *'node_data/siomonit_file*

```
# Sample `node_data/siomonit_file
#
#  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
#
# up to three siologin arg lists like the ones above may be included, one
#  for each SIO line (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
```

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

```
/sys/siologin/siologin  siologin_arg_list
```

Special Considerations

Use the logs in *‘node_data/siomonit_log* and *‘node_data/siologin_log*. They can help you debug **siologin** or **siomonit** server problems. Table 6–8 shows an example of an *siomonit_log*.

Table 6–8. Example *‘node_data/siomonit_log*

```
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 *
```

Often, the failure of an **siologin** process to stay alive can be caused by a locked SIO line (*/dev/siox*). 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 your *‘node_data/startup_sio.sh* file.

If **siomonit** terminates and its child processes do not also terminate, the child processes are, in effect, orphans. The existence of the orphans will interfere with **siomonit**'s attempts to restart new child processes when it restarts. **Siomonit** itself never stops a child process, its own or an orphan. 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.

spm – 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 on line. **Spm** 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 using the shell command **crp** with options similar to the Display Manager's **cpo** and **cps** commands. The process you create may run another server, such as **prsvr** or **netman**. The process can also run a shell program.

- Log in to the node for debugging purposes or to maintain servers. For example, you might want to use the shell Command **sigp** to stop a server process running on the node.

Starting and Stopping spm

From the DM command line;

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

The process begins immediately and continues after log out.

From the *'node_data/startup[suffix]* file, remove the # sign from the following line:

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

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

```
            % sigp server_process_manager -q
```

New at SR9.5 – shutspm

You can also use the **shutspm** command to shut down the server process manager on a remote server node. When the **spm** runs in place of the DM, it waits on the eventcount file *'node_data/spmshut_ec.* The **shutspm** command advances this eventcount, causing the **spm** to perform an orderly shutdown of the node.

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

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.%       pgndwrx
  %.%.%               dr
```

This ACL limits **shutspm** shutdown to *sys_admin* login 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.

To prevent **spm** from responding to the **shutspm** command, add the following line to the *'node_data/ startup.spm* file:

```
  no_shutspm
```

Tablet Server

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 node's *'node_data/startup[suffix]* file. The server process sends a control character to the tablet bit pad and to the **–bp_enable** option of the shell command **tctl**. An operating system program then provides the actual bit pad support. The process started in the *'node_data/startup[suffix]* file stops running when the operating system has taken control.

Starting the Tablet Server

Remove the "#" sign from following line in the *'node_data/startup[suffix]*, or *'node_data/startup.spm* files:

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

To use a different SIO line, change the "2" in the command to the desired number. The "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 **pst** 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
```

DOMAIN/IX Servers

Many of the network server processes that are peculiar to DOMAIN/IX are discussed later in this book. See Chapter 7 for information about **inetd** (the super-daemon and the daemons it controls), **tcp_server, routed, rwhod,** and Chapter 10 for details about running **sendmail** as a server process. Chapter 7 also includes instructions for invoking these daemons, and the circumstances under which you wish to run them. Several other servers that are not discussed elsewhere are explained below.

The writed daemon

The **write**(1) command allows users to communicate interactively by writing messages into a small window on each other's displays. Although the decision whether to run **writed** is generally made at the node level, rather than the network level, we've included information about it here, as well as in Chapter 11's collection of administrative commands. In order for a node to accept a **write** message, it must be running the **writed**(8c) daemon process, as well as the */sys/mbx/mbx_helper* process. The **writed** server is usually invoked in the per-node file *'node_data/etc.rc*. The default entry, supplied at DOMAIN/IX installation, reads:

```
# if [-f /etc/writed ] ; then
#         /etc/writed &
# fi
```

Remove the pound signs (#) from the beginning of all three lines, and when the **/etc/run_rc** process is started, the **writed** will start, too. The **writed** process starts an **mbx_helper** automatically if one is not already running on the node.

See the *DOMAIN/IX Command Reference for BSD4.2* for information on the **write** command itself.

The talkd daemon

DOMAIN/IX includes the **talk** user program, which operates in a similar fashion to the **write** program. For any node to receive a **talk** message, it node must be running the */etc/talkd* daemon, which is enabled in the *'node_data/etc.rc* file, and started by the **run_rc** process. An entry in *'node_data/etc.rc* of the following form, if uncommented by removing the # characters, will start */etc/talkd* at boot time.

```
# if [-f /etc/talkd ] ; then
#         /etc/talkd &
# fi
```

In order to accept incoming **talk** requests, the destination node must be correctly configured to run DOMAIN TCP/IP. See Chapter 7 for details. The **talkd** daemon listens at the 'udp' socket specified in the **talk** services description. See **services**(5) for further information on services descriptions.

The cron daemon

There are certain commands, like **crpasswd** and **sendmail,** that you may wish to execute on a regular basis. The **/etc/cron** daemon executes commands at specified dates and times according to formatted instructions in a per-node file called */usr/lib/crontab*. Under DOMAIN/IX *bsd4.2*, the */usr/lib/crontab* file is actually a link to the node's */sys/cron/crontab* file. Normally, the */sys/cron/crontab* file on the system administrator's node will contain scheduling information and system administration commands to be run by **cron**. The */usr/lib/crontab* (i.e., */sys/cron/crontab*) on a user's node should not run any system administration commands; it should be reserved to schedule user-level commands only for that node.

The necessary lines to run the **cron** command are included in each user's *'node_data/etc.rc* file; uncomment them to start **cron** at login time. An example of the format of */usr/lib/crontab* for a system administrator node follows. Note the commands to run such processes as **crpasswd** and the **uucp** commands, which should be reserved to the system administrator node's */usr/lib/crontab* file.

```
0 6 * * * /etc/crpasswd
5 * * * * /com/date >>/sys/node_data/syslog
30 19,20,21,22,23,0,1,2,3,4,5,6,7 * * * /usr/lib/uucp/uu.hourly
0 3 * * * /usr/lib/uucp/uu.daily
0 1 * * 1 /0 2 1 * * /usr/lib/uucp/uu.monthly
45 0,1,2,3,4,5,6,12,17,19,20,21,22,23 * * * /usr/lib/uucp/uuxqt.sh
7,37 * * * * /usr/lib/sendmail -q
```

The first five fields of *crontab* specify the minute, hour, day of the month, month of the year, and day of the week. The asterisks (*) signify that **cron** should execute that line's entry at all legal values of that particular field. (For a list of all legal values for all fields, see the manual page in Chapter 11 for **cron**.) The sixth field in *crontab* is the command that the shell is to execute at the given time or times. For example, **cron** interprets the first line of our sample *crontab* file this way: run the **/etc/crpasswd** program at 6:00 AM every day. Running **crpasswd** every day ensures that, even if you've added new accounts during the day, the mapping between DOMAIN **person** and **project uids** and DOMAIN/IX **user** and **group** IDs will always be up–to–date by the next day.

The second line places the output of the **/com/date** program into the file */sys/node_data/syslog* at five minutes past every hour, and so on. The **cron** program looks at the *crontab* file once a minute to see if there are commands in it for **cron** to execute.

Chapter 7

Configuring TCP/IP

The version of DOMAIN TCP/IP available at SR9.5 includes a new feature, a subnet utility that allows you to divide an internet at your site into sub–internets, without having to create a unique network entry in the DARPA Internet routing tables for each subnet. The major difference in configuring TCP/IP is in the addition of a *subnet mask*, which changes the interpretation of the standard Internet address. This version of TCP/IP also includes newer versions of **htable**(8) and **routed**(8c) that support subnetting. For complete information, see the Release Notes that accompany Version 3.0 of the TCP/IP product.

Defining the TCP/IP Configuration

Before you can use TCP/IP, you must decide a number of configuration issues that affect how the TCP/IP software establishes connections between hosts. You will have to define both addressing information, which defines the names, addresses, and physical interfaces for the hosts, networks, and gateways, and process information, which specifies the processes that you run to support TCP/IP communications and the nodes on which these processes execute.

This part of the chapter defines this information, and explains its functions. It also defines steps you should take before you install TCP/IP that will make the configuration procedure easier.

Names and Addresses

Whenever you refer to a TCP/IP host, you usually use an easy to remember name, such as the node name //dionysus. The operating system converts this name internally to an address value that is meaningful to it, for example the DOMAIN address 03a2c176.06d49.

Within a DOMAIN network, this mapping between names and addresses is sufficient to get messages from one point to another. However, when you connect different local networks, such as DOMAIN and ETHERNET, each network's local addresses are meaningless to the next. Therefore, another addressing layer is required; in the case of connecting to an ETHERNET, this is the Internet addressing layer.

While local addresses need only be unique on the local network, Internet addresses must be unique across all connected networks. For example, let's give the node //dionysus the Internet address 197.9.8.3.

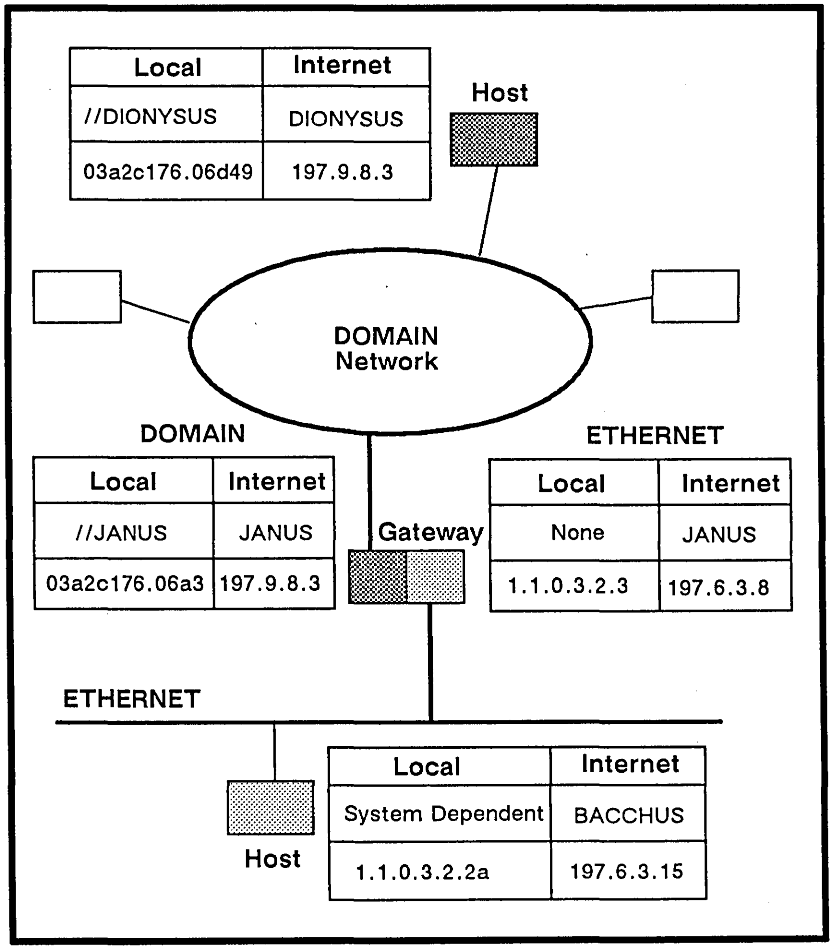
TCP/IP also uses mnemonic names. To keep things simple, the local name is usually the Internet name. While DOMAIN/IX TCP/IP does not require that the node name and Internet name be the same, we recommend that you follow this convention for the sake of simplicity.

Gateway Names and Addresses

While a host has one local name and address and one Internet name and address, a gateway must have more than one address, because it sits on two local networks and must be fully identified on both. For example, the DOMAIN node //janus is a gateway to an ETHERNET network. It has:

- The DOMAIN name //janus
- The DOMAIN address 03a2c176.06a3
- The Internet address on the DOMAIN network 197.9.8.1
- The Internet name janus
- The Internet address on the ETHERNET LAN 197.10.9.1
- The ETHERNET address 1.1.0.3.2.3

Figure 7–1 shows the network example that we have been using. It shows a DOMAIN network and ETHERNET LAN connected to make an Internet. It illustrates the names and addresses for a DOMAIN host, gateway, and an ETHERNET host.



Local	Internet
//DIONYSUS	DIONYSUS
03a2c176.06d49	197.9.8.3

DOMAIN

Local	Internet
//JANUS	JANUS
03a2c176.06a3	197.9.8.3

ETHERNET

Local	Internet
None	JANUS
1.1.0.3.2.3	197.6.3.8

Local	Internet
System Dependent	BACCHUS
1.1.0.3.2.2a	197.6.3.15

Figure 7–1. Relationships Among Names and Addresses

Names

Since people generally prefer to use names rather than Internet addresses, TCP/IP requires you to associate an ASCII name with each host and gateway with which you communicate. While TCP/IP does not require you to do so, people usually use the node or computer names used in their local network as the TCP/IP names. For example, if your node entry name is //janus, you'd name the host janus.

The Internet model allows you to use more than one name for a host or gateway. The additional names are called aliases. Therefore, you could give //janus not just the Internet name janus, but also the Internet aliases odin and thor.

Internet names for a gateway do not depend upon the local network. Therefore, if janus is a gateway between a DOMAIN network and an ETHERNET LAN, you can use janus, odin, or thor from hosts on either network.

Local names depend upon the operating system that runs on the host. Because of the design of the DOMAIN system, you use DOMAIN names on a network-wide basis. This is not necessarily true on the ETHERNET LAN, where one host may be a VAX running UNIX and another host may run a completely different operating system.

Local Network Addresses

A DOMAIN address consists of one or two parts, depending upon whether you have a multiple-network DOMAIN internet.

If your installation consists of a single network, your DOMAIN address may consist of a 20-bit **Node ID**. This number is firmware-encoded for each DOMAIN node and always appears as a hexadecimal number with up to five digits. For example, the node ID for //janus is 006a3. You can determine any node's ID by using the **/com/netstat** Shell command.

If your installation uses the DOMAIN/BRIDGE to connect multiple networks, a **Network number** precedes the Node ID and is separated from the node ID by a period. The network number is an up-to eight-digit hexadecimal number that identifies the host's network. Network numbers are assigned by the Apollo Response Center. In our example, //janus is on a network with a network number of 03a2c176. Therefore, //janus' full DOMAIN address is 03a2c176.6a3.

An ETHERNET address is a 48-bit number. Three bytes are reserved for the manufacturer's identifiers. Three bytes are assigned by the XEROX Corporation.

Internet Addresses

Each Internet address is 32 bits long, and contains two variable-length fields that identify the local network and the host within that network. The network number, the left field, identifies the local network to the Internet. The host number, the right field, identifies the host within the local network.

You normally represent Internet addresses in **Internet format**. In the Internet format, you specify addresses as follows:

```
W.X.Y.Z
```

where W, X, Y and Z are decimal numbers between 0 and 255. Each of the decimal numbers represents one byte in the Internet address.

Internet addresses have variable-length fields for the network numbers and host numbers. You choose a length for your network numbers by choosing Type A, B, or C Internet addresses. Figure 7-2 illustrates the differences. It also shows how the most significant bits (MSB) in each network number serve as tag bits to identify the address type as A (MSB of 0), B (MSB of 10) or C (MSB of 110).

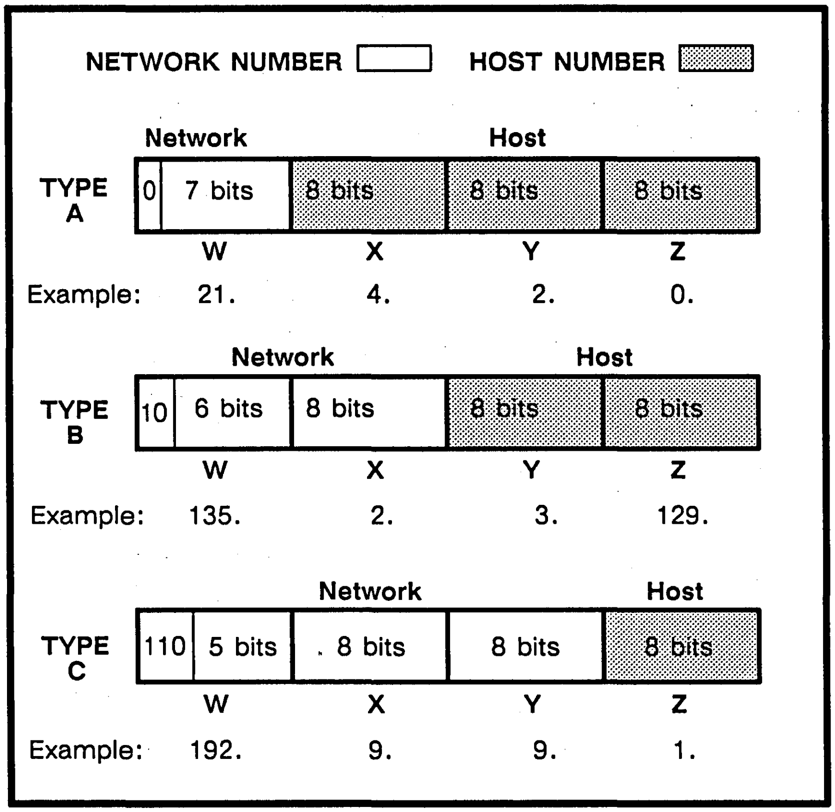


Figure 7-2. Type A, B, and C Internet Addresses

For example, the Type A address in the figure has a network number of 21. Contrast this with the network number in the Type B address (135.2) and in the Type C address (192.9.9). In the Type C address, you may only assign host numbers 0 – 255, whereas in the Type A address, you can assign host numbers 0.0.0 through 255.255.255. We discuss considerations for assigning Internet addresses later in this appendix.

Physical Layer Interface Descriptions

The Internet model's layered approach allows communication between networks with varied Physical Layer protocols. The hardware that supports DOMAIN TCP/IP-based communications translates protocols between the ETHERNET LAN and the DOMAIN network. You must indicate to TCP/IP the **physical interface** used by each host and by the gateway.

For hosts in the DOMAIN network, the interface is the DOMAIN network. The DOMAIN network always has physical interface identifier of **dr0**. Gateways have two physical interfaces; the DOMAIN network and the ETHERNET LAN. These are identified as **dr0** and **il0**, respectively.

A DOMAIN node that is a router for the DOMAIN BRIDGE is treated as a TCP/IP gateway node. Its physical interfaces are **dr0**, for the DOMAIN network on which it resides, and **dr1**, for the DOMAIN network to which it is connecting via TCP/IP. Like any other DOMAIN TCP/IP gateway, a DOMAIN BRIDGE router must run the **tcp_server** process and **routed**.

Daemons, Servers, and Helpers

DOMAIN/IX TCP/IP-based communications software uses several processes that support communications in various ways. These processes respond to requests for some form of service. They are generally called **servers** or **daemons**. Table 7-1 lists these processes, describes their purposes, and indicates the nodes that require them.

Table 7-1. DOMAIN/IX TCP/IP Server Processes

Name	Purpose	Location
tcp_server	Enables TCP/IP communications on the node	All TCP/IP hosts
routed	Manages gateway network routing tables	One per gateway node
rwhod	System status server, maintains database used by **rwho** and **ruptime**	One per gateway node
sendmail	Handles mail received over the Internet	One per network
tftpd	Enables **tftp** access to the host	Hosts that accept **tftp** connections
inetd	Starts the following daemons as needed. (in */etc/inetd.conf*)	All *bsd4.2* nodes
ftpd	Enables direct **ftp** access to the host	Hosts that accept **ftp** connections
telnetd	Enables inbound **telnet** access to the host	Hosts that accept inbound **telnet**
rexecd	Enables remote execution of commands on this node	Hosts that accept **rexec** routine
rlogind	Enables remote login to this node	Hosts that accept **rlogin** program
rshd	Enables remote execution of commands on this node with user authentication	Hosts that accept **rsh** program

Starting Server Processes

As a general rule, these processes should execute whenever the node is running; therefore you include commands to start them in startup files. The files you use depend upon the processes.

Configuring Servers and Helpers

You start some of the server processes in the node startup file. The name and location of this file depends upon the type of node that you are using. If you are configuring a disked node, use the */sys/node_data/ startup[.node_type]* file. If you are configuring a diskless node, use the *//partner_node/sys/*

node_data.nodeid/startup[.node_type] file. In these pathnames, *.node_type* indicates the model node you are using; *.nodeid* is the node identification number. Chapter 3 describes node startup files in detail.

You use the node startup file to specify the following process:

- **tcp_server**

To specify the **tcp_server**, uncomment or enter the following line:

```
cps /sys/tcp/tcp_server -n tcp_server
```

Configuring the *bsd4.2* Daemons

NOTE: In most installations, the */etc* directory is located on a single administrative node, and all other nodes use links to access this directory. In such cases, */etc/rc* on the administrative node *must* be a link to the file *‘node_data/etc.rc*, and */etc/inetd.conf must* be a link to *‘node_data/etc.inetd.conf*. The *‘node_data* is automatically interpreted as either */sys/node_data* or *//partner_node/sys/node_data.nodeid* of the node that is accessing the file.

The *bsd4.2* daemons are started by the **/etc/run_rc** program, which starts the daemons specified in the */etc/rc* file. You start **/etc/run_rc** at login time, in the node's startup file. Put the following line in either the */sys/node_data/startup[.node_type]* or *//partner_node/sys/node_data.nodeid/startup[.node_type]* file.

```
cps /etc/run_rc
```

Use the */etc/rc* file to specify any of the following processes:

- **inetd**
- **routed**
- **rwhod**
- **sendmail**
- **tftpd**

Use the */etc/inetd.conf* file to specify the following processes. Be certain that there are no blank lines in the */etc/inetd.conf* file, as this may cause the process to fail. (You must, of course, also specify **inetd** in the */etc/rc* file.)

- **ftpd**
- **rexecd**
- **rlogind**
- **rshd**
- **telnetd**

Specify each required daemon by uncommenting the lines in the file that start the process. For example, to specify **inetd**, uncomment (remove the # characters from the beginning of) the following lines in */etc/rc*:

```
if [ -f  /etc/inetd ]; then
             /etc/inetd &
fi
```

The tcp_server

The **tcp_server** process ensures that all TCP/IP data are reliably transmitted between the end-user processes such as **ftp** and **telnet**. It also performs routing services, and maintains mapping tables that relate Internet addresses to local addresses.

Rules: The **tcp_server** must run on each node that uses any form of TCP/IP based communications.

bsd4.2 Daemons

You must run the *bsd4.2* daemons to enable various *bsd4.2* commands and utilities. The following sections describe these daemons and indicate the nodes where they should execute. See the *DOMAIN/IX Programmer's Reference for bsd4.2* for detailed descriptions of each daemon.

routed

The **routed** daemon maintains the gateway network routing tables. It maintains an internal database of gateways that are directly accessible from the local gateway. It also broadcasts its routing tables every 30 seconds. This dynamic operation eliminates the need to update static gateway tables each time the network configuration changes.

If **routed** does not execute on the gateway, the gateway does not transmit packets with routing information to the ETHERNET. Because **routed** purges its database of gateway entries for gateways that have not sent a network information packet within three minutes, remote *bsd4.2* gateways that run **routed** may lose knowledge of any DOMAIN gateway that does not use **routed.**

Rules: **routed** must execute on each gateway.

routed is not required if you only use *bsd4.2* TCP/IP communications on the DOMAIN network.

rwhod

The **rwhod** daemon is the Internet system status server. It maintains the database of status information that is used by the **rwho**(1) and **ruptime**(1) programs.

Rules: **rwhod** should execute on each gateway. In this case, it can provide information on both the DOMAIN network and the ETHERNET.

sendmail

The **sendmail** program routes mail messages that you send using *bsd4.2* or DARPA mail commands over the Internet. When it runs as a daemon, it enables the DOMAIN network to send and receive **mail** messages from the ETHERNET. **Sendmail** is not a user-level interface.

Rules: If you use mail between the DOMAIN network and any other network, **sendmail** must run as a daemon on one node on your network. You execute **sendmail** as a daemon by including the **-bd** flag in the **sendmail** command.

tftpd

The **tftpd** daemon is a server that supports the DARPA Trivial File Transfer Protocol. It listens for and accepts **tftp** requests. You can establish a **tftp** request only to a node that runs the **tftpd**. That is, you must specify a host that runs **tftpd** in the Shell **tftp** command. However, the **tftpd** daemon provides access to files on all nodes on the DOMAIN network. You do not need to run **tftpd** to issue the **tftp** command.

Rules: You should have one or more **tftpd** processes per DOMAIN network if you wish to support the TFTP protocol.

inetd

The **inetd** is a server-manager that invokes Internet services such as **ftpd** or **rlogind** as necessary. Since it is a single process, **inetd** can efficiently manage many types of Internet connections

You must use **inetd** to invoke the following servers and daemons:

- **ftpd**
- **rexecd**
- **rlogind**
- **rshd**
- **telnetd**

Rules: You must have an **inetd** process on each node that requires **ftpd, rexecd, rlogind, rshd,** or **telnetd.**

The file **/etc/inetd.conf** specifies the daemons that **inetd** invokes. You only include those daemons in this file for the services that are supported by the server node. A template file, located in */sys/node_data/ etc.inetd.conf,* is automatically installed with DOMAIN/IX *bsd4.2.* You include daemons by removing the comment marks (#) at the beginning of the lines that specify the required daemons. Be certain that there are no blank lines in the */etc/inetd.conf* file, as this may cause the process to fail.

ftpd

The **ftpd** daemon accepts **ftp** connections and services **ftp** requests. You can establish an **ftp** connection only to a node that can run the **ftpd.** That is, you must specify a node that can run **ftpd** in the Shell **ftp** command or in response to the **ftp Host:** prompt. However, the **ftpd** provides access to files on all nodes on the DOMAIN network. You do *not* need the **ftpd** to issue the **ftp** command.

Rules: You should have one or more **ftpd** processes per DOMAIN network.

You must use the **inetd** daemon to invoke the **ftpd.**

ftpd and the DOMAIN FTP_SERVER cannot execute on the same node.

rexecd

The **rexecd** daemon services requests from the **rexec**(3X) library function. It allows you to execute DOMAIN/IX commands remotely, on the server node. **Rexecd** must receive a valid user id and password from **rexec.**

Rules: **rexecd** must execute on each node that supports invocation of commands from a remote host that uses **rexec.**

You must use the **inetd** daemon to invoke the **rexecd.**

rlogind

The **rlogind** daemon services requests from the **rlogin**(1) program. It allows you to log in remotely on the server node. **Rlogind** requires pseudo-ttys. These are normally created when you install DOMAIN/IX, but can be created by the **crpty**(8) command. **Rlogind** does not request a password, if the remote host is listed in the server's */etc/hosts.equiv* file.

Rules: **rlogind** must execute on each node that supports login from a remote host using **rlogin.**

You must use the **inetd** daemon to invoke the **rlogind.**

rshd

The **rshd** daemon is the remote shell server. It services requests from the **rsh** program and **rcmd** library function. It allows you to remotely execute DOMAIN/IX commands on the server node. **Rshd** does not request a password, if the remote host is listed in the server's */etc/hosts.equiv* file.

Rules: **rshd** must execute on each node that supports invocation of commands from remote hosts using **rsh** or **rcmd**.

You must use the **inetd** daemon to invoke the **rshd**.

telnetd

The **telnetd** daemon accepts **telnet** connections. You must run the **telnetd** on each node that accepts inbound **telnet** sessions. You do not need the **telnetd** to issue the **telnet** command.

Rules: **telnetd** must execute on each node that accepts remote login using **telnet**.

You must use the **inetd** daemon to invoke the **telnetd**.

telnetd and the DOMAIN TELNET_SERVER can not execute on the same node.

Service Nodes

Before we can discuss TCP/IP Mapping information files, we must introduce an additional concept that is not part of standard TCP/IP, but reflects the distributed nature of the DOMAIN system. While TCP/IP communications distinguishes between gateways and hosts, we add a third type of node, the service node, for the purposes of configuration.

Defined broadly, a service node is any node that provides some form of support service to other users. A service node does not necessarily have to use TCP/IP communications itself. Conversely, a service node can also be either a host or a gateway. The TCP/IP service node is that node which contains the TCP/IP name and address mapping files.

The DOMAIN/IX administrative node, which contains the */etc* directory, is also considered a service node, since the *bsd4.2* mapping files are located in that directory. The TCP/IP service node and the DOMAIN/IX administrative node may be the same machine.

TCP/IP Mapping Information Files

You provide the information about a host or gateway's names, addresses, and physical interfaces, and about the relations among them during the configuration process by editing several files. However, before you edit the files you should understand the contents of each file. This section describes each file, and the following sections will help you prepare for configuration. Table 7–2 lists the files that you edit and indicates their functions and locations.

Note that there are two files (*/etc/networks* and */etc/hosts*) that you must manage only if you are using DOMAIN/IX *bsd4.2* TCP/IP on a DOMAIN network that does *not* communicate with any other networks. These files are automatically generated (by **makehost.sh**) if you are using *bsd4.2* and communicating across a gateway to another Internet network, but if you are using *bsd4.2* TCP/IP on a DOMAIN network only, you must edit them manually.

Table 7–2. TCP/IP Information Files that you Edit

Name	Purpose	Location
On all Hosts		
thishost	Lists the Internet name of the local host.	*/sys/node_data [node.id]/thishost*
networks	Lists the Internet addresses and corresponding physical interfaces for the local host.	*/sys/node_data [.nodeid] /networks*
On the Service Node		
local.txt	Contains information on locally defined Internet addresses.	*/sys/tcp/hostmap/ local.txt*
hosts.txt	Contains information on Internet addresses defined by the Network Information Center (NIC).	*/sys/tcp/hostmap/ hosts.txt*
On the Gateway Node		
host_addr	Associates Internet and local addresses of non–DOMAIN hosts that do not use the Address Resolution Protocol (ARP).	*/sys/tcp/host_addr*
On the *bsd4.2* Administrative Node		
hosts.equiv	Contains the names of all hosts that you can access using **rlogin, rsh** and **rcp** without password authentication.	*/etc/hosts.equiv*
On the *bsd4.2* Administrative Node if There is No Gateway		
networks	Contains the Internet network number of the DOMAIN network. This file is different from the */sys/node_data[node.id]/networks* file.	*/etc/networks*
hosts	Contains the names and Internet addresses of all hosts on the DOMAIN network	*/etc/hosts*

Links and File Locations

The service node files and the *bsd4.2* configuration files usually are located on only one or a few nodes on the DOMAIN network. You access them through links from each host and gateway. This technique limits the replication of information, and therefore the number of changes you must make when the network configuration changes. It also provides a measure of security.

When you install DOMAIN/IX *bsd4.2* or TCP/IP on a host node, the install procedure creates the required links between your node and the TCP/IP service node and *bsd4.2* administrative node. However, you should be familiar with these links, so we list them in Table 7-3.

Table 7-3. TCP/IP Links

Pathname on Host	**Links to:**
On all Hosts	
/sys/tcp/hostmap	*//service_node/sys/tcp/hostmap* (directory)
/sys/tcp/gateways	*//service_node/sys/tcp/gateways*
/sys/tcp/hosts.hst	*//service_node/sys/tcp/hosts.hst*
/etc (directory)	*//administrative_node/etc*
On *bsd4.2* Administrative Nodes	
/etc/rc	*'node_data/etc.rc*
/etc/inetd.conf	*'node_data/etc.inetd.conf*

Some Notes on Pathnames

The following notes describe how we refer to certain files in this manual.

- In this manual we refer to pathnames in the */sys* directory. This is a relative pathname, and is accurate only if you are working at the node you are configuring. If you are manipulating the */sys* directory of a remote node you must use an absolute pathname. For a disked node this is *//nodename/sys*; for a diskless node it is *//partner_node/sys* (where *partner_node* is the node name of the diskless node's partner).
- Files in the */sys/node_data* directory are specific to the node on which the directory is located. Therefore, the corresponding files for a diskless node are located in a special directory, *//partner_node/sys/node_data.nodeid*, where *nodeid* is the node number of the diskless node. We refer to these directories by using the convention */sys/node_data[.nodeid]*
- The pathname *'node_data* is a relative pathname. It always refers to the node_data directory of the node that is making the reference.

THISHOST File

The */sys/node_data[.nodeid]/thishost* file defines the Internet name of the local host. You must have a *thishost* file on each node that is a TCP/IP host, including the gateway. The file consists of the host's Internet name on a single line. For example, the *thishost* file for `janus` is simply:

```
janus
```

NETWORKS File

The *networks* file defines the Internet addresses and the physical interface name of the local host. On disked nodes, it is located in */sys/node_data/networks*. On diskless nodes it is located in */sys/*

node_data.nodeid/networks, where *nodeid* is the node number of the diskless node. You must have a *networks* file for each node that is a TCP/IP host, including the gateways. The *networks* file format is:

internet_address ON physical_interface [; comment]

A host node's *networks* file consists of a single line with the host's Internet address on the DOMAIN network and the physical interface identifier **dr0**. A gateway's *networks* file also must have the gateway's Internet address on the ETHERNET, with the interface identifier **il0**. For example, the *networks* file for the gateway //janus looks like:

```
197.9.8.1 on dr0
197.10.9.1 on il0
```

The physical interface for the second DOMAIN network that a DOMAIN BRIDGE router is connected to is represented as **dr1**. If //janus were a DOMAIN BRIDGE router, its *networks* file would look like this:

```
197.9.8.1 on dr0
X.X.X.X   on dr1
```

where X.X.X.X was the Internet address for janus on the other DOMAIN network.

Service Node Configuration Files

The *hosts.txt* File

The */sys/tcp/hostmap/hosts.txt* file is the official Department of Defense Internet Host table from the Network Information Center (NIC). This file contains the names and addresses of all the hosts on the ARPANET, as well as many other networks in the Internet. You must have a copy of this file on your service node if you will communicate over ARPANET or any other network that is listed by the NIC.

If do not plan to connect your DOMAIN network to the DoD Internet, you should replace *hosts.txt* with an empty file or rename the file. Doing so will make the Shell script **makehost.sh** run faster.

We include a copy of *hosts.txt* with the TCP/IP software. However, the copy we send may not be the most current copy of the *hosts.txt* file. We suggest that you use the copy we ship you initially to create your host table, and then retrieve the latest version by executing the **gettable**(8) program. See the *DOMAIN/IX Programmer's Reference for bsd4.2* for a detailed description of this utility.

The *local.txt* File

The */sys/tcp/hostmap/local.txt* file contains network information that is not provided by the Network Information Center in the *hosts.txt* file. You should use this file, and not *hosts.txt* if you assign your own network number and addresses. You usually do this if your hosts do not use any network listed by the Network Information Center. You should also add to this file the names and Internet addresses of new computers and networks that are not yet in the NIC–supplied version of *hosts.txt*.

To add these names and Internet addresses, edit the version on the service node. You'll keep only one version of the *local.txt* and *hosts.txt* file in your network, on the service node. Use the following information when you edit the *local.txt* file.

NOTE: The format of the *local.txt* file is the same as the format for the *hosts.txt* file. The format of these files is defined in RFC 810, "DoD Internet Host Table Specification" and is available online at NIC as the file:

[SRI-NIC]<NETINFO>RFC810.TXT

You can retrieve this file through **ftp** using username ANONYMOUS with any password. See *Using telnet and ftp* for information about **ftp**.

The *local.txt* file has three categories of entries called NET, GATEWAY, and HOST.

- The **NET** entries define the networks that you can access. You list the network number and a name for each network linked in your TCP/IP implementation. For DOMAIN TCP/IP implementations, you'll be listing a DOMAIN network (or, if you use DOMAIN/BRIDGE connections, a DOMAIN Internet) and an ETHERNET LAN.

- The **GATEWAY** entries specify the gateways between the networks that you can access. In this entry or entries, you list both Internet addresses of each gateway node, its name, and certain other information.

- The **HOST** entries specify the TCP/IP hosts that you can access. In these entries, you list each host, its name, and other information. You have as many HOST entries as there are hosts that you wish to access, including remote hosts. (If you know the remote hosts are listed in *hosts.txt*, you need not add them to *local.txt*; however, it's fine to have entries listed in both files.)

```
; NET : NET-ADDR : NETNAME :
; GATEWAY : ADDR, ADDR : NAME : CPUTYPE : OPSYS : PROTOCOLS:
; HOST : ADDR, ALTERNATE-ADDR (if any): HOSTNAME,NICKNAME:
;   CPUTYPE : OPSYS : PROTOCOLS :
;
NET : 197.6.3.0 : OFFICE-ETHER :
NET : 197.9.8.0 : DOMAIN-RING :
GATEWAY : 197.9.8.1,197.6.3.8 : JANUS, APOLLO : DN460 : AEGIS : IP/GW, GW/DUMB :
; The following are remote hosts on the ETHERNET
HOST : 197.6.3.15 : BACCHUS : VAX/11-750 : VMS : TCP/TELNET, TCP/FTP :
HOST : 197.6.3.18 : ODIN, WOTAN : VAX/8650 : UNIX : TCP/TELNET, TCP/FTP :
HOST : 197.6.3.22 : TESTBED : VAX/11-785 : UNIX :TCP/TELNET, TCP/FTP :
; The next host entry is for the gateway node. (The gateway
; needs a host entry. also, since it's also a host.)
HOST : 197.9.8.1, 197.6.3.8 : JANUS, : DN460 : DOMAIN : TCP/TELNET,TCP/FTP :
; The rest of the file lists hosts on the DOMAIN network.
HOST : 197.9.8.3 : DIONYSUS : DN550 : DOMAIN/IX : TCP/TELNET, TCP/FTP:
HOST : 197.9.8.5 : PAN : DN460 : DOMAIN/IX : TCP/TELNET,TCP/FTP :
HOST : 197.9.8.8 : ATHENA : DN300 : DOMAIN/IX : TCP/TELNET, TCP/FTP:
HOST : 197.9.8.9 : APOLLO : DN460 : DOMAIN/IX : TCP/TELNET, TCP/FTP:
```

Figure 7-3. A local.txt File

Each entry consists of the keyword **NET, GATEWAY,** or **HOST** followed by from two to five fields. You should start the *local.txt* file with all NET entries, followed by all GATEWAY entries, and then all HOST entries.

The following rules apply to each entry:

- A semicolon (:) terminates each field.

- Double colons (::) indicate a null field.

- A comma (,) separates values within a field

- A semicolon (;) begins a comment. Any text on a given line following the semicolon is not part of the host table. A comment can follow an entry on the same line.

- Spaces are optional before and after commas and colons.

Each **NET** entry in the *local.txt* file has the following format:

NET : net–addr : netname :

Where:

net–addr is the network's Internet network number, followed by a 0; for example 197.9.8.0

netname is the name of the network; for example OFFICE–ETHER.

Each **GATEWAY** entry in the *local.txt* file has the following format:

```
GATEWAY : addr1,addr2 : name : [cputype :] [opsys :]
   [protocols :]
```

Where:

addr1 is the address of the gateway on one of the networks that it connects; for example 197.9.8.1

addr2 is the address of the gateway on the other network that it connects; for example 197.6.3.8

name is the Internet name of the gateway; for example, `janus`.

cputype This optional field describes the gateway processor. This field provides you with information; TCP/IP software does not use it. To ensure consistency you should use workstation model numbers for DOMAIN nodes, for example DN330.

opsys This optional field describes the gateway processor's operating system. This field provides you with information; TCP/IP software does not use it. To ensure consistency, you should use **DOMAIN/IX** for nodes that only run DOMAIN/IX and **AEGIS/DOMAIN/IX** for nodes that run both.

protocols This optional field describes the Internet protocols that the gateway supports. For DOMAIN/ETHERNET Gateways, specify:

IP/GW Internet Gateway

and either:

GW/DUMB Non–routing gateway, DOMAIN gateways are normally non–routing.

or:

GW/PRIME Prime gateway. Specify this protocol *only* if you wish to access hosts on other networks that are connected (directly or indirectly) to the ETHERNET network. *Do not* specify both GW/PRIME and GW/DUMB.

Each **HOST** entry in the *local.txt* file has the following format:

```
HOST : addr [,alt-addr] : name [,nickname] : [cputype :]
   [opsys :] [protocols :]
```

Where:

addr is the Internet address of the host; for example, 197.9.8.5

alt-addr is one or more alternate Internet addresses for the host, separated by commas; for example, 197.6.3.8

name is the Internet name of the host; for example, odin.

nickname is one or more alternate names that you can use to access the host. All names must be separated by commas; for example, wotan, snaer.

cputype This optional field describes the host processor. This field provides you with information; TCP/IP software does not use it. To ensure consistency, you should use workstation model numbers for DOMAIN nodes, for example, DN330.

opsys This optional field describes the host operating system. This field provides you with information; TCP/IP software does not use it. To ensure consistency, you should use **DOMAIN/IX** for nodes that run DOMAIN/IX and **AEGIS/DOMAIN/IX** for nodes that run both.

protocols This optional field describes the Internet protocols that the host or gateway supports. This field provides you with information; TCP/IP software does not use it. For DOMAIN hosts, including gateways, specify both of the following. If your node supports other protocols, specify them as defined in RFC 810.

TCP/FTP	FTP file transfer protocol
TCP/TELNET	Telnet terminal emulator protocol

HOST_ADDR File

If you will be communicating with any systems on the ETHERNET LAN that do not support the Address Resolution Protocol (ARP), you must put information about the system's ETHERNET address in the */sys/tcp/host_addr* file and run the **maphost** program on the gateway (from the */sys/node_data[node.id]/startup[node_type]* file, after the **tcp_server** is started. See *Managing TCP/IP-Based Communications Products* for details.

DOMAIN/IX Files

DOMAIN/IX *bsd4.2* uses the following files:

- */etc/hosts.equiv*
- */etc/networks*
- */etc/hosts*

You must create the */etc/networks* and */etc/hosts* files only if you are using DOMAIN/IX *bsd4.2* TCP/IP–based communications on a DOMAIN network that does not have a gateway. You must have these files if you use **rlogin, lpr, rcp, rsh,** and **rexec,** as well as **ftp** and **telnet.** You do not have to create these files on DOMAIN networks that use a TCP/IP gateway, because the **/sys/tcp/hostmap/makehost.sh** Shell script creates them from information in the *local.txt* file.

hosts.equiv

The */etc/hosts.equiv* file lists hosts that are equivalent to your host for login purposes. That is, if a host is on the */etc/hosts.equiv* file for you node it can execute any of the following programs or functions using your node:

- **lpr**(1)
- **lprm**(1)
- **lpq**(1)

- **rcmd**(3X)
- **rcp**(1)
- **rlogin**(1) (without entering a password)
- **rsh**(1)

Note that the node that runs the line printer daemon lpd must be configured for TCP/IP communications and must have the names of all nodes that will print files using this daemon in its */etc/hosts.equiv* file.

The */etc/hosts.equiv* file contains the name of each equivalent TCP/IP host, one name per line. For example:

```
janus
dionysys
bacchus
pan
athena
```

networks File

The */etc/networks* file consists of a single line with the following form for each Internet network:

```
domain-ring            network_number
```

Where **domain-ring** is a keyword (all lower case) and **network_number** is the Internet network number of the DOMAIN network, and the two values are separated by one or more blanks or TAB characters.

In a DOMAIN network where there is no gateway to additional (non-DOMAIN) networks, this file consists of a single line. For example:

```
domain-ring  197.9.8
```

The */etc/hosts* File

The */etc/hosts* file contains the name and Internet address of each TCP/IP host. Each line has the following form:

```
Internet_address       Host_name
```

The address and name must be separated by one or more blanks or TAB characters.

For example:

```
197.9.8.1           janus
197.9.8.3           dionysus
197.9.8.5           pan
197.9.8.8           athena
197.9.8.9           apollo
```

Defining the Configuration

This section describes how you can define the configuration for a DOMAIN network that uses TCP/IP communications. You can skip this information and begin the configuration procedure if you are familiar with TCP/IP or if you are configuring a single node.

To define the configuration you:

1. Select the network, gateway and host Internet Addresses.
2. Define the host mapping files.

3. Select the nodes that execute special server processes.

Selecting Internet Addresses

You must select Internet addresses for your hosts and gateways before you can configure TCP/IP–based communications.

- You must assign an Internet address to each node that contains TCP/IP software and acts as a host. All the Internet addresses you assign to hosts on a DOMAIN network *must* have the same network number, but different host numbers.

- You must assign two internet addresses to a gateway node. Since the gateway node belongs to two local networks, its two Internet addresses have different network numbers.

As described before, Internet addresses have variable–length fields for the network numbers and host numbers. You choose a length for your network numbers by choosing Type A, B, or C Internet addresses. Type A, B, and C Internet addresses differ in the size of the network number and host number fields. Table 7–4 summarizes these differences:

Table 7–4. Type A, B, and C Internet Address Comparison

Type	Format		Description	Example
A	W. (network number)	X.Y.Z (host number)	One–byte network number, three–byte host number.	48.1.2.1 (decimal)
B	W.X. (network number)	Y.Z (host number)	Two–byte network number, two–byte host number.	139.2.9.2 (decimal)
C	W.X.Y. (network number)	Z (host number)	Three–byte network number, one–byte host number.	192.9.1.2 (decimal)

For example, the Type A address in the table has a network number of 48. Contrast this with the network number in the Type B address (139.9) and in the Type C address (192.9.1). In the Type C address, you may only assign host numbers 0 – 255, whereas in the Type A address, you can assign host numbers 0.0.0 through 255.255.255.

You should choose an address type, network numbers, and host numbers to suit the number of hosts at your site. If you plan to use DOMAIN TCP/IP to communicate on the ARPANET, you must apply to the Network Information Center (NIC) for a network number.

Use these guidelines to select Type A, B, or C addressing:

- Type A addresses allow for large numbers of hosts, up to 16,777,216, on a single network. Choose Type A addressing ONLY if you plan never to access the ARPANET, because many Type A addresses are already assigned by the Network Information Center (NIC).

  Type A network numbers must be in the range 0 – 127. Type A host numbers must be in the range 0.0.0 through 255.255.255. Therefore, Type A addresses range from 0.0.0.0 through 127.255.255.255.

- Type B addresses allow for up to 65,535 hosts. Choose Type B addresses if you plan a very large number of hosts. The NIC will sometimes assign you a Type B network number.

Type B network numbers must be in the range 128.0 – 191.255. Type B host numbers must be in the range 0.0 through 255.255. Therefore, Type B addresses range from 128.0.0.0 through 191.255.255.255.

- Type C addresses allow for only 256 hosts, but allow network numbers between 192.0.0 and 255.255.255 (three–byte network numbers). Therefore, Type C addresses range from 192.0.0.0 through 255.255.255.255. If you apply to the NIC for a network number, you will usually receive a Type C network number.

In most cases, you should use Type C addresses even if you don't plan to register with NIC and access the ARPANET immediately. We recommend Type C addresses because you can choose a number not yet used by any other network. (To see the reserved network numbers, read the file */sys/tcp/hostmap/hosts.txt*.) Then, if you wish to access the ARPANET in the future, you will not need to change network numbers in your Internet addresses.

Defining the Mapping Files

Once you have defined the Internet addresses, you can determine the location of the network-wide mapping files that you will need, and their contents. You should:

1. Determine the service node (or nodes) that will have the host mapping tables.

2. Determine the name of the *bsd4.2* administrative node (or nodes) that will have the */etc* directory.

3. If you are configuring TCP/IP that uses a gateway, define the contents of the *//service_node/sys/ tcp/ local.txt.*

   If you are configuring *bsd4.2* TCP/IP on a network that does not have an ETHERNET gateway, define the contents of the */etc/networks* and */etc/hosts* files.

Determining the Service and Administrative Nodes

You may wish to have one or a few service nodes. If you have a single such node, then you only have to maintain a single database. If you have a large network or wish to ensure the availability of mapping information, additional nodes can be helpful. However, each host is linked to a single node, and you will have to manually change the links in order to change the host's service node. Also, you must update each service node whenever mapping information changes. The same considerations apply to selecting administative nodes.

The service node does not have to be a TCP/IP host or gateway. Because the DOMAIN/IX administrative node uses *bsd4.2*, it is usually a TCP/IP host.

Defining the *local.txt* file

Define the *local.txt* file by determining the information required for each network, gateway, and host in your internet. If you allow connection to the DARPA internet, do not include any information that is already in the *hosts.txt* file.

If you are connecting the DOMAIN network to an Internet, that is to more than just a single ETHERNET, you should apply the following considerations when you define the *local.txt* GATEWAYS entries.

- The order that you use for the GATEWAY entries affects the gateway that is used when TCP/IP software establishes a connection. TCP/IP always tries to use the first applicable gateway on the list.

Defining the */etc* files

You need to define the */etc* directory files if you are configuring a DOMAIN/IX *bsd4.2* network that does not use gateways. In this case, the */etc/networks* file consist of a single line with the network name and address, for example,

```
our-net-ether        197.9.8
```

The */etc/hosts* file must contain the Internet address and name of each *bsd4.2* node on the network.

Determining Server Processes

Before you begin configuring a network that uses TCP/IP you should determine which nodes use which server processes.

Configuring TCP/IP

This part describes how to configure a DOMAIN network or node that uses TCP/IP communications or supports TCP/IP based communications. It also briefly describes requirements for configuring TCP/IP on non-DOMAIN hosts.

If you are configuring a DOMAIN network to use TCP/IP-based communications for the first time, you must configure all of the nodes that use or support TCP/IP communications.

Configuring TCP/IP on an Internet

If you are configuring a DOMAIN network that uses an ETHERNET gateway to communicate with other networks, use the procedures described in this appendix in the following order:

1. Use Procedure 7-1 to configure the service node or nodes.
2. Use Procedure 7-2 to configure each node that uses DOMAIN/IX TCP/IP, that is, each host and gateway.
3. Use Procedure 7-4 to configure the remote (non-DOMAIN/IX) hosts that will communicate with the DOMAIN network.

Configuring TCP/IP on a DOMAIN Bridge Network

A bridge network is a physical link that connects two DOMAIN networks. A bridge router node is a node configured with internet communications hardware and connected to a bridge network. If your installation uses *bsd4.2* TCP software to handle internet routing, every bridge router node must be running **routed**(8C); every node in the network must have an *etc.rc* entry of the form:

```
/etc/route add bridgenet router 1
```

where *bridgenet* is the name of the bridge network to which *router* is a gateway. If the network includes multiple routers, there must be an entry like this for every router. The information written by **route** is only meaningful if */sys/tcp/tcp_server* is running. Since both the **tcp_server** and the **run_rc**(8) program are typically started by entries in *'node_data/startup[.type]*, it is important that the line

```
cps /sys/tcp/tcp_server
```

precede the *etc/run_rc* line in the startup file.

Configuring DOMAIN-Only *bsd4.2* TCP/IP

If you are configuring a DOMAIN network that supports DOMAIN/IX *bsd4.2*, but does not have access to any non-DOMAIN networks, use procedure 3 to configure TCP/IP on each *bsd4.2* node.

Configuring DOMAIN/IX Nodes

For purposes of installing and configuring TCP/IP, DOMAIN/IX nodes can be divided into three classes:

Host	Nodes that use TCP/IP communications to communicate with other hosts
Gateway	Nodes that connect two networks (Gateways are also TCP/IP hosts)
Service	Nodes that aid TCP/IP communications but do not necessarily act as hosts. Service nodes provide host mapping tables and contain the */etc* directory.

Additionally, the exact procedures that you use depend on whether your node:

- Has a disk or is diskless
- Is a DOMAIN/IX *bsd4.2* node on a DOMAIN network that is not connected to any other networks.

This part of the appendix includes procedures for all types of nodes.

Table 7–5 lists the procedures in this part and types of hosts that they can configure. The rest of this part consists of the configuration procedures.

NOTES: These procedures refer to the */sys/node_data[.nodeid]* and *'node_data* directories. These two terms are not interchangeable; the first is an absolute reference, the second is relative. Using the wrong name in a link can result in circular file references.

Always configure the service node before you configure hosts and gateways. This order insures that all tables are up-to-date and eliminates any duplication of effort

Table 7–5. Configuration Procedures

Procedure	*System*	*Type*
1		Service node
2	DOMAIN/IX	Internet Host or Gateway
3	DOMAIN/IX	Host on a single DOMAIN network
4	Various	Remote Host

All of these procedures describe the steps required for either a disked or diskless node. (However, you should not have a diskless service node, as the node's main function is to provide mapping files information.) Similarly, procedures 2 and 3 describe procedures for hosts and gateways. Some steps in these procedures are required for only disked, or only diskless nodes or for gateways only, and they are marked as such.

PROCEDURE 7-1. Configuring the Service Node

NOTE: This procedure assumes that you are using a DOMAIN/IX *bsd4.2* C Shell.

□ Task 1: Select the Internet Addresses

If you have not already done so, determine the Internet addresses for all hosts, including the remote hosts that you can access across the gateway.

□ Task 2: Install the Software

If you have not already done so, use the procedures described in the associated Release Notes to install software in the following order:

NOTES: See the TCP/IP Release Notes to determine the revision level required for each of the following software.

If you are configuring a diskless host, you must install the following software on the hosts's partner node.

1. DOMAIN

2. C, DOMAIN/IX

   Note: When you install *bsd4.2* DOMAIN/IX you must give **/etc/run_rc** root ownership. Otherwise, processes required for TCP/IP will not execute properly.

3. TCP/IP

   A service node can be a host or gateway, but does not have to be either. The files that are installed when you install TCP/IP depend upon the options you select.

   The Release Notes list the files are installed in each case. Use the **ls** command to make sure that all the required files are installed. For example, if this node is a service node only:

```
% ls /sys/tcp
host_addr    hostmap
% ls /sys/tcp/hostmap
hashnic      hosts./txt   htable      local.txt
makehdb      makehost.sh  sortnic
```

□ Task 3: Configure the hosts.txt File

If you are connecting your DOMAIN network to a Department of Defense (DoD) Internet, then you should use a */sys/tcp/hostmap/ hosts.txt* file. This file contains the Internet mapping information for all the hosts, gateways, and networks on the ARPANET and other networks on the Internet.

NOTE: The copy of *hosts.txt* that we supply may not be the most current version available. After you have configured at least one host, you can obtain the current *hosts.txt* file and update the mapping tables by running the **gettable**(8) command.

If you are *not* connecting your DOMAIN network to the DoD Internet, replace this file with an empty file. If you wish to save the information, rename the file to */sys/tcp/hostmap/hosts.txt.SRn*, where *n* is the software release number. Using an empty file prevents the configuration procedure from adding information in the *hosts.txt* file to the mapping tables, and speeds up Task 5.

□ Task 4: Configure the local.txt File

Edit the */sys/tcp/hostmap/local.txt* file.

1. Add a **NET** entry to the file for *each* network that you can access, including your DOMAIN network. Do not add any entries that are already in the *hosts.txt* file. The **NET** entries should be the first entries in the file. **NET** entries have the following format:

   NET : netaddr : netname

   Where netaddr is the network's Internet network number followed by **.0**, **.0.0**, or **.0.0.0**, depending on whether the address is Type A, B, or C, and netname is the name that you have assinged to the network.

   For example, if your DOMAIN network has an Internet network number of 197.9.8 and is named SAMPLE-NET, enter the following in the *local.txt* file:

   ```
   NET : 197.9.8.0 : SAMPLE-NET
   ```

2. Add a **GATEWAY** entry to the file for *each* gateway between networks that you can access. Do not add any entries that are already in the **hosts.txt** file. All **GATEWAY** entrys should follow the **NET** entrys in the file. **GATEWAY** entries have the following format:

   GATEWAY : addr1, addr2 : name : [cputype :] [opsys :] [protocols :]

   Where addr1 is the gateway's Internet address on one network, addr2 is the gateway's Internet address on the second network, and name is the gateway's host name.

   For example, if your gateway is named janus, is a DSP90 running AEGIS, and has an Internet address of 197.9.8.1 on the DOMAIN network and an address of 197.6.3.8 on the ETHERNET, enter the following in the local.txt file:

   ```
   GATEWAY : 197.9.8.1, 197.6.3.8 : JANUS : DSP90 : DOMAIN : IP/GW , GW/DUMB
   ```

3. Add a **HOST** entry to the file for each host *and* each gateway on *all* networks that you can access. Do not add any entries that are already in the **hosts.txt** file. All **HOST** entries should follow the **GATEWAY** entrys in the file. **HOST** entries have the following format:

   HOST : addr[,alt-addr] : name[,nickname] : [cputype :] [opsys :] [protocols :]

   For example, if a host is named dionysus, is a DN550 running AEGIS, and has an Internet address of 197.9.8.3, enter the following in the local.txt file:

   ```
   HOST : 197.9.8.3 : DIONYSUS : DN550 : DOMAIN :
     TCP/TELNET, TCP/FTP:
   ```

❑ Task 5: Create the Host Tables

Run the **/sys/tcp/hostmap/makehost.sh** Shell script from any node. This script converts the *local.txt* file into a format that TCP/IP software can use.

For example:

```
% /sys/tcp/hostmap/makehost.sh
Formatting tables
Sorting Tables
Formatting tables (pass 2)
Hashing
input: hosts.tmp1, 1466 lines
output: hosts.hst, 215519 bytes
hash: 1542 bytes, 257 entries, 0 holes
keys: 9794 bytes 534 entries (largest 662 longest 37)
data: 116019 bytes, 1466 entries (largest 740)
(file) "hosts.hst" moved.
(file) "gateways" moved.
(file) "hosts.tmp" deleted.
(file) "hosts.tmp1" deleted.
(file) "nets.tmp" deleted.
(file) "hosts.txt1" deleted.
(file) "nets.txt1" deleted.
```

□ Task 6: : Edit the /etc/hosts.equiv File

The */etc/hosts.equiv* may or may not be physically located on the service node. We include the file here because it performs a service function. Also, if you edit */etc/hosts.equiv* at this point when you configure a network for the first time, you do not have to edit it when you configure individual hosts.

As a general rule, the */etc* directory resides on the DOMAIN/IX administrative node and all other nodes access it through links. In most cases, only the system administrator may be able to edit this file.

Enter the name of each host that you will allow to execute the **rsh**(1) program or **rcmd**(3) routine, or to use **rlogin**(1) without entering a password in the */etc/hosts.equiv* file. Each entry in this file consists of a single line with the host name, for example:

```
lilly
```

□ Task 7: Configure Other Services

The service node can also be a TCP/IP host or gateway. You should now configure that facility.

- If the Service is a TCP/IP host or gateway, continue with Procedure 7–2, skipping steps 1 through 4.

END OF PROCEDURE 7–1.

Procedure 7-2. Configuring A DOMAIN/IX *bsd4.2* Host or Gateway Node

□ Task 1: Select an Internet Address

If you have not already done so, select an Internet address for the host.

□ Task 2: Stop the tcp_server

If you are already using TCP/IP on the node, find the process ID of the **tcp_server** and stop that process by entering the following commands:

```
% ps aux
USER      PID      SZ        STAT      TIME       COMMAND

root      2        0         R         7693:05    null
root      34       0         S         3425       tcp_server
.
.
.

% kill -9 34
```

□ Task 3: Install the Software

If you have not already done so, use the procedures described in the associated Release Notes to install software in the following order:

NOTES: See the appropriate Release Notes to determine the revision level required for each of the following software.

If you are configuring a diskless host, you must install the following software on the host's partner node.

1. DOMAIN
2. C, DOMAIN/IX *bsd4.2*

   Note: When you install the *bsd4.2* you must give **/etc/run_rc** root ownership. Otherwise, processes required for TCP/IP will not execute properly.

3. TCP/IP

   After you install TCP/IP, execute the **ls** command to check that the TCP/IP software was properly installed and that the four required links from the */sys/tcp* directory to the mapping service node exist. In the following example, **edny** is the mapping service node:

```
% ls -l /sys/tcp
total 452
-rwxrwxrwx  1 root    9824 Sep 19 13:42 ether_diag
-rwxrw-rwx  1 root   78664 Sep 19 13:42 ftp_server
lrwxrwxrwx  1 root      27 Jan  8 10:06 gateways -> //edny/sys/tcp/gateways
lrwxrwxrwx  1 root      28 Jan  8 10:06 host_addr ->//edny/sys/tcp/host_addr
lrwxrwxrwx  1 root      26 Jan  8 10:06 hostmap  -> //edny/sys/tcp/hostmap
lrwxrwxrwx  1 root      28 Jan  8 10:06 hosts.hst-> //edny/sys/tcp/hosts.hst
drwxrwxrwx  1 root    1024 Sep 19 13:42 lib
-rwxrwxrwx  1 root   27050 Sep 19 13:42 makegate
-rwxrwxrwx  1 root    3122 Sep 19 13:42 maphost
-rwxrwxrwx  1 root      19 Sep 19 13:57 networks
-rwxrwxrwx  1 root  103214 Sep 19 13:42 tcp_server
-rwxrwxrwx  1 root   91640 Sep 19 11:02 tcp_server.sr8.gateway
-rwxrwxrwx  1 root    4722 Sep 19 13:42 tcpinit
-rwxrwxrwx  1 root    1662 Sep 19 11:02 tcpreset
-rwxr-xr-x  1 root   21648 Dec 17 15:27 telnet_server
-rwxrwxrwx  1 root       7 Sep 19 13:56 thishost
```

□ Task 4: Update the Mapping Service Node Host Tables

NOTE: If you are configuring TCP/IP for the first time on your network you should configure your mapping service nodes before you configure your host nodes. The mapping service node files will then be up-to-date. If you've already configured your service nodes, skip this task.

If you are configuring a single node for the first time, or if you are changing the node's Internet address, use the following steps to update the TCP/IP mapping service node's host mapping tables:

NOTE: The install procedure creates the pathname */sys/tcp/hostmap* on your local node as a link to the directory *//mapping_service_node/sys/tcp/hostmap*.

1. Add (or change) the node's **HOST** entry in the */sys/tcp/hostmap/local.txt* file. Each host *and* gateway must have an entry with the following format:

   ```
   HOST : addr[,alt-addr] : name[,nickname] : [cputype :] [opsys :] [protocols :]
   ```

   For example, if your host is named dionysus, is a DN550 running DOMAIN/IX, and has an Internet address of 197.9.8.3, enter the following in the *local.txt* file:

   ```
   HOST : 197.9.8.3 : DIONYSUS : DN550 : DOMAIN/IX : TCP/TELNET, TCP/FTP:
   ```

2. **Gateways Case:** If you are configuring a gateway it must *also* have a **GATEWAY** entry in the *local.txt* file. All **GATEWAY** entries must precede the **HOST** entries in the file.

   ```
   GATEWAY : addr1, addr2 : name : [cputype :] [opsys :] [protocols :]
   ```

   Where addr1 is the gateway's Internet address on one network, and addr2 is the gateway's Internet address on the second network.

   For example, if your gateway is named janus, is a DSP90 running DOMAIN/IX, and has an Internet address of 197.9.8.1 on the DOMAIN network and an address of 197.6.3.8 on the ETHERNET, enter the following in the *local.txt* file:

   ```
   GATEWAY : 197.9.8.1, 197.6.3.8 : JANUS : DSP90 :
        DOMAIN/IX : IP/GW , GW/DUMB
   ```

3. Run the **/sys/tcp/hostmap/makehost.sh** Shell script from any node. This script converts the *local.txt* file into a format that TCP/IP software can use.

For example:

```
% /sys/tcp/hostmap/makehost.sh
Formatting tables
Sorting Tables
Formatting tables (pass 2)
Hashing
input: hosts.tmp1, 1466 lines
output: hosts.hst, 215519 bytes
hash: 1542 bytes, 257 entries, 0 holes
keys: 9794 bytes 534 entries (largest 662 longest 37)
data: 116019 bytes, 1466 entries (largest 740)
(file) "hosts.hst" moved.
(file) "gateways" moved.
Updating 4.2bsd host tables
(file) "/etc/hosts" moved.
(file) "/etc/gateways" moved.
(file) "/etc/networks" moved.
(file) "hosts.tmp" deleted.
(file) "hosts.tmp1" deleted.
(file) "nets.tmp" deleted.
(file) "hosts.txt1" deleted.
(file) "nets.txt1" deleted.
```

□ Task 5: Update the /etc/hosts.equiv File

NOTE: In most installations the */etc* directory resides on an administrative node and all other nodes access it through links. In this case, only the system administrator may be able to edit this file.

If you will allow this host to execute the **rsh**(1) program or **rcmd**(3) routine, or to use **rlogin**(1) without entering a password, then enter the host name in the */etc/hosts.equiv* file. Each entry in this file consists of a single line with the host name, for example:

```
lilly
```

□ Task 6: Edit the Node Startup Files

Use the following steps to update your node startup files.

1. Edit the */sys/node_data[.node_id]/startup[.type]* file to include the following commands:

   REQUIRED: These commands *must* be in the following order. Otherwise, processes that are initialized by **run_rc** will not execute. However, you can include other commands between them.

   ```
   cps /sys/tcp/tcp_server –n tcp_server
   env SYSTYPE  'bsd4.2'
   cps /etc/run_rc
   ```

2. Edit the */etc/rc* file by removing the comment character (#) from the lines that contain processes that you want to run on this host. The "*bsd4.2* Daemons" section describes the processes used for TCP/IP–based communications in detail.

   In most cases you will want to run **inetd**, which can start several TCP/IP–related daemons. In this case, uncomment the following lines:

   ```
   if [ –f /etc/inetd ]; then
       /etc/inetd &
   fi
   ```

3. If you specified **inetd** in step 2, edit the */etc/inetd.conf* file by removing the comment character (#) from the lines that contain processes that you want to run on this host. Be certain that there are no blank lines in the */etc/inetd.conf* file, as this may cause the process to fail.

❏ Task 7: Edit the Shell Login Files

Edit your **.cshrc** file if you use the C shell or your **.profile** file if you use the Bourne shell to include the /com directory in the search path. The */com* directory includes the **tcpstat, host, net,** and **edip** commands that you use to monitor and manage TCP/IP communications. By including the */com* directory in the search path you eliminate the need to specify the full pathname for these commands.

The */com* directory also includes the DOMAIN versions (as opposed to *bsd4.2* versions) of **telnet** and **ftp**. Therefore, the */com* directory must follow the */usr/ucb* directory in the search path.

For example, include the following line in your **.cshrc** file:

```
set path=(. /bin /usr/bin /usr/ucb /com )
```

or include the following line in your **.profile** file:

```
PATH=:.:/bin:/usr/bin:/usr/ucb:/com:)
export PATH
```

❏ Task 8: Edit the thishost File

Disked Case: If your node has a disk, edit the */sys/node_data /thishost* file to replace the word "apollo" with your host's name. This file must consist of a single line with your host's name, for example:

```
lilly
```

Diskless Case: If your node is diskless, copy the partner node's */sys/node_data/thishost* file to */sys/node_data.node_id /thishost*. Then edit this file to replace the host name with your host's name.

❏ Task 9: Edit the networks File

Edit the */sys/node_data[.node_id]/networks* file to include the host's Internet address and physical interface.

This file looks as follows immediately after you install TCP/IP:

```
0.0.0.0 on dr0
0.0.0.0 on il0
```

Diskless Case: If your node is diskless, copy the partner node's */sys/node_data/networks* file to */sys/node_data.node_id /networks*. Then continue with the **Host** or **Gateway** case.

Host Case: Hosts have a single address and a single physical interface. The host's physical interface must be **dr0**. Therefore, a host's *networks* file must have a single entry.

Edit the file by changing the address on the first line and deleting the second line. For example, if your host's Internet address is 197.9.8.3, you should edit the *networks* file to look as follows:

```
197.9.8.3 on dr0
```

Gateway Case: Gateways have one address and physical interface for each network. The physical interface for the DOMAIN network is **dr0**; the physical interface for the ETHERNET network is **il0**. Therefore, a host's *networks* file must have two single entries.

Edit the file by changing the address on each line. For example, if your gateway's Internet address on the DOMAIN network is 197.9.8.1 and the address on the ETHERNET LAN is 197.6.3.8, you should edit the *networks* file to look as follows:

```
197.9.8.1 on dr0
197.6.3.8 on il0
```

□ Task 10: Initialize TCP/IP

You initialize TCP/IP by starting the node's **tcp_server** process and updating its host tables, if necessary.

1. Start the **tcp_server**.

   - Reboot the node. Because you have included the required command in the node startup file, the **tcp_server** automatically initiallizes when the node reboots. Rebooting the node also ensures that any other TCP/IP processes are started, and that all processes are up-to-date.

     To reboot the node:

     1. Enter the DM **ex** command as follows:

        ```
        Command: ex
        ```

        All current processes stop executing, the operating system exits, and the node enters the bootshell, which prompts you with parentheses. Enter the **go** command as follows:

        ```
        ) go
        ```

        AEGIS reboots and returns you to the DM login message. You can now log in and use TCP/IP.

   NOTE: When the **tcp_server** initializes it automatically executes the following programs:

   - **/sys/tcp/tcpinit**
   - **/sys/tcp/makegate**

2. If your node will communicate with non-DOMAIN hosts that do not understand the Address Resolution Protocol (ARP), run the **/sys/tcp/maphost** program.

END OF PROCEDURE 7-2.

PROCEDURE 7-3. Configuring a DOMAIN/IX *bsd4.2* Host that Communicates Only On the DOMAIN Network

□ Task 1: Select an Internet Address

If you have not already done so, select an Internet address for the host.

□ Task 2: Stop the tcp_server

If you are already using TCP/IP on the node, find the process ID of the **tcp_server** and stop that process by entering the following commands:

```
% ps aux
USER      PID       SZ        STAT      TIME      COMMAND

root      2         0         R         7693:05   null
root      34        0         S         3425      tcp_server
.
.
.

% kill -9 34
```

□ Task 3: Install the Software

If you have not already done so, use the procedures described in the associated Release Notes to install software in the following order:

NOTES: See the appropriate Release Notes to determine the revision level required for each of the following software.

If you are configuring a diskless host, you must install the following software on the hosts's partner node.

1. DOMAIN
2. C
3. DOMAIN/IX *bsd4.2*

   Note: When you install *bsd4.2* you must give **/etc/run_rc** root ownership. Otherwise, processes required for TCP/IP will not execute properly.

□ Task 4: Update the /etc Directory files

If you are configuring a single node for the first time, or if you are changing the node's Internet address, use the following steps to update the TCP/IP files in the */etc* directory:

NOTE: In most installations the */etc* directory resides on an administrative node and all other nodes access it through links. In this case, only the system administrator may be able to edit these files.

1. Add an entry to the */etc/hosts* file. Each entry consists of a single line with the Internet address followed by the host name. For example, if your host's Internet Address is 197.9.8.3 and it's name is `dionysus`, add the following line to the */etc/hosts* file:

   ```
   197.9.8.3        dionysus
   ```

2. Make sure there is an entry in the */etc/networks* file. There should only be a single line in this file, consisting of a network name followed by the Internet network number. Therefore, you only need to edit this file *once*, when you configure the first host on your network.

For example, if your network is named **sample-ring** and your network number is 197.9.8, the */etc/networks* file should look as follows:

```
sample-ring        197.9.8
```

3. If you will allow this host to execute the **rsh**(1) program or **rcmd**(3) routine, or to use **rlogin**(1) without entering a password, then enter the host name in the */etc/hosts.equiv* file. Each entry in this file consists of a single line with the host name, for example:

```
lilly
```

❑ Task 5: Edit the Node Startup Files

Use the following steps to update your node startup files.

1. Edit the */sys/node_data[.node_id]/startup[.type]* file to include the following commands. The commands must be in the following order, but you can include other commands between them.

```
cps /sys/tcp/tcp_server -n tcp_server
env SYSTYPE  'bsd4.2'
cps /etc/run_rc
```

2. Edit the */etc/rc* file by removing the comment character (#) from the lines that contain processes that you want to run on this host. The "*bsd4.2* Daemons" describes the processes used for TCP/IP-based communications in detail.

   In most cases you will want to run **inetd**, which can start several TCP/IP-related daemons. In this case, uncomment the following lines:

```
if [ -f /etc/inetd ]; then
    /etc/inetd &
fi
```

3. If you specified **inetd** in step 2, edit the */etc/inetd.conf* file by removing the comment character (#) from the lines that contain processes that you want to run on this host. Be certain that there are no blank lines in the */etc/inetd.conf* file, as this may cause the process to fail.

❑ Task 6: Edit the Shell Login Files

Edit your **.cshrc** file if you use the C shell, or your **.profile** file if you use the Bourne shell, to include the */com* directory in the search path. The */com* directory includes the **tcpstat, host, net,** and **edip** commands that you use to monitor and manage TCP/IP communications. By including the */com* directory in the search path you eliminate the need to specify the directory in these commands.

The */com* directory also includes the DOMAIN versions (as opposed to *bsd4.2* versions) of **telnet** and **ftp**. Therefore, the */com* directory must follow the */usr/ucb* directory in the search path.

For example, include the following line in your **.cshrc** file:

```
set path=(. /bin /usr/bin /usr/ucb /com)
```

or include the following line in your **.profile** file:

```
PATH=:.:/bin:/usr/bin:/usr/ucb:/com:
export PATH
```

❑ Task 7: Edit the thishost File

Edit the */sys/node_data[.node_id]/thishost* file to replace the word "apollo" with your host's name. This file must consist of a single line with your host's name, for example:

lilly

□ Task 8: Edit the networks File

Edit the */sys/node_data[.node_id]/networks* file to include the host's Internet address and physical interface. The physical interface for a host that is not a gateway must be **dr0**.

This file looks like this immediately after you install TCP/IP:

```
0.0.0.0 on dr0
0.0.0.0 on il0
```

If your Internet address 197.9.8.3, you should edit this file to look as follows:

```
197.9.8.3 on dr0
```

□ Task 9: Initialize TCP/IP

You initialize TCP/IP by starting the node's **tcp_server** process.

- Reboot the node. Because you have included the required command in the node startup file, the **tcp_server** automatically initiallizes when the node reboots. Rebooting the node also ensures that any other TCP/IP server processes are started, and that all processes are up-to-date.

  To reboot the node:

  1. Enter the DM **ex** command as follows:

     ```
     Command: ex
     ```

     All current processes stop executing, the operating system exits, and the node enters the bootshell, which prompts you with parentheses.

  2. Enter the **go** command as follows:

     ```
     ) go
     ```

     AEGIS reboots and returns you to the DM login message. You can now log in and use TCP/IP.

NOTE: When the **tcp_server** initializes it automatically executes the following programs:

- */sys/tcp/tcpinit*
- */sys/tcp/makegate*

END OF PROCEDURE 7-3.

Configuring Non–DOMAIN Hosts

In the same way that you must add the names and Internet addresses of foreign networks, gateways, and hosts to the *local.txt* file on the DOMAIN network, you must add the names and Internet addresses of the DOMAIN network, gateway, and hosts to some equivalent file or files on ETHERNET side of the connection. Each foreign host that will communicate with the DOMAIN network must have access to this information.

We can not provide procedures for all possible situations; however, we *can* provide some general rules that may be helpful if you are a first–time user of TCP/IP. The following sections cover the case if you are using a DARPA standard TCP/IP implementation and if you are using *bsd4.2* UNIX TCP/IP.

Configuring DARPA TCP/IP Hosts

If you are configuring a host that uses DARPA standard TCP/IP you must edit the *hosts.txt* or an equivalent file used for hosts that are not listed by the NIC to include the DOMAIN network, gateway, and hosts. After you edit this file you may have to execute additional steps to update the host's internal mapping tables.

Configuring *bsd4.2* UNIX Hosts

If you are configuring a host that uses *bsd4.2* UNIX you must do the steps described in Procedure 7–2 at the non–DOMAIN host. We assume several things in this procedure, including:

- You have the permissions required to edit such files as */etc/hosts* on the non–DOMAIN host.
- You understand the */etc/rc* file on the UNIX system well enough to set up the servers and daemons that are appropriate for the UNIX host.

PROCEDURE 7-4. Configuring a Non-DOMAIN *bsd4.2* Host to Communicate with a Host on a DOMAIN Network

□ Task 1: Edit /etc/networks

Make sure that there is an entry in the */etc/networks* file for the DOMAIN network that you want to access. Unless you are adding the first DOMAIN host, there should already be an entry in the file. The */etc/ networks* file entries consist of the network's name followed by its Internet network number. For example:

```
domain-ring2          197.9.8
```

Note that in this example the three digit Internet network number indicates that the DOMAIN network uses class C Internet addresses.

The method you use to manage the */etc/networks* file depends upon your system administartion procedures. For example, you might edit the */etc/networks* file directly. Or, you might edit a file similar to *local.txt* file and use the **htable**(8) command to put the entries in the */etc/networks* file. See your host's *bsd4.2* documenation for more information on **htable**.

□ Task 2: Edit /etc/hosts

Make sure that there is an entry in the */etc/hosts* file for the DOMAIN host (or hosts) that will communicate with this host. Entries in the */etc/hosts* file entries consist of the network's name followed by its Internet network number. For example:

```
197.9.8.3             dionysus
```

The method you use to manage the */etc/hosts* file depends upon your system administration procedures. For example, you might edit the */etc/hosts* file directly. Or, you might edit a file similar to *local.txt* file and use the **htable**(8) program to put the entries in the */etc/hosts* file. See your host's *bsd4.2* documentation for more information on **htable**.

□ Task 3: Edit /etc/hosts.equiv

Enter in the */etc/hosts.equiv* file the name of each DOMAIN host that you will allow this host to access using the **rsh**(1) program or **rcmd**(3) routine, or to use **rlogin**(1) without entering a password. Each entry in this file consists of a single line with the host name, for example:

```
dionysus
```

□ Task 4: Edit /etc/rc

Edit the non-DOMAIN host's */etc/rc* file, if necessary. This file controls the services that this host provides to other hosts; it is a shell script that executes automatically when the remote UNIX system is rebooted. Some installations use a */etc/rc.local* file for commands that are pertinent to a single site. For more details on **rc** see **rc**(8) in the *UNIX Programmer's Manual for bsd4.2.*

The */etc/rc* file must specify the **routed** routing daemon and any other daemons, such as **telnetd** and **ftpd**, that you require to enable TCP/IP communications with the DOMAIN hosts.

□ Task 5: Ensure that the daemons are Executing

Make sure the routing daemon, **routed**, and any other daemons that you require for TCP/IP communications with DOMAIN hosts run on this host. You can check whether this process is running by using the **ps** command. If necessary, reboot the system or start the processes manually.

Chapter 8

Line Printer Management

This document describes the structure and installation procedure for the line printer spooling system developed for the *bsd4.2* version of DOMAIN/IX.

The line printer system supports:

- multiple printers,
- multiple spooling queues
- local and remote printers, and
- 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 aprinter 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)

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.

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 *'node_data/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 data base 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.

The DOMAIN/IX version of the *printcap* file has one additional entry, which is explained more fully on the **printcap**(5) *DOMAIN/IX Programmer's Reference Manual* pages. It is **pc**, which provides an interface to print commands you can use instead of sending output to **lp** or **rp**. In DOMAIN/IX, this is set to use the DOMAIN **/com/prf** command sequence.

Prerequisites for DOMAIN/IX

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.

To set up **lpd** to run at boot time, uncomment (remove the # from the beginning of) the three lines in the *'node_data/etc.rc* file that read:

```
#if [ -f /usr/lib/lpd ]; then
#       /usr/lib/lpd &
#fi
```

Shut down the node and restart it. The DOMAIN/IX 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 *DOMAIN/IX Programmer's 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 below for references to information about DOMAIN/IX TCP/IP.)

The DOMAIN/IX implementation of the line printer spooler runs over a TCP/IP connection. Therefore, DOMAIN/IX TCP/IP must be configured on all machines that are to use the **lp** system, and the DOMAIN *tcp_server* must be running on all nodes. See Chapter 7 for information on configuring TCP/IP correctly.

Note that the node to which the printer is physically connected does not have to have either TCP/IP or the *tcp_server* running, unless that node is also going to run the **lp** commands or the **lpd** process.

Commands

All of these commands, their options, and any arguments are explained fully on the *DOMAIN/IX Command Reference Manual* pages, and the manual pages for system administration that are included in Chapter 11 of this volume.

lpd - line printer daemon

The program **lpd**(8), usually invoked at boot time from the *'node_data/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 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. **Lpd** 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 table 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 a 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. **Lpd** 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.

lpq - show line printer queue

The **lpq**(1) program works recursively backwards, displaying the queue of the machine directly connected to the printer and then the queue(s) of the machine(s) that lead to it. **Lpq** has two forms of output. In the default short format, it gives a single line of output per queued job (first example below). In the long format (second example), it shows the list of files which comprise a job, and their sizes.

```
% lpq
Rank  Owner        Job    Files                   Total Size

1st   cass         18     memo, manual 4997 bytes
```

```
% lpq -l
Warning: no daemon present

cass: 1st                    [job 018apollo]
      memo                    456 bytes
      manual                 4541 bytes
```

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 filenames.

lprm - remove jobs from a queue

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.

lpc - line printer control program

The **lpc**(8) program is used by the system administrator to control 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.

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 insures 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. **Lpd** 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*.

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 DOMAIN/IX implementation of **lp** supports only the **lp** printer filter.)

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 which 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.

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 login 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.

Output filter specifications

The filters supplied with DOMAIN/IX *bsd4.2* handle printing and accounting for printers supported by the the DOMAIN print command and print server, **/com/prf** and **/com/prsvr**, respectively. 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 cleanup 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 -wwidth -llength
```

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] -wwidth -llength -iindent -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 -xwidth -ylength -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.

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).

abort and start

Abort 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 (i.e., **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). **Abort** only operates on a machine that is 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.

Start enables printing and requests **lpd** to start printing jobs.

enable and disable

Enable and **disable** 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.

restart

Restart allows you to restart printer daemons when **lpq** reports that there is no daemon present.

stop

Stop is used to halt a spooling daemon after the current job completes; this also disables printing. This is a clean way to shutdown a printer in order to perform maintenance, etc. Note that users can still enter jobs in a spool queue while a printer is stopped.

topq

Topq 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.

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, the name printer is the name of the printer. This would be one of the names from the */etc/printcap* database.

lpr

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 which 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 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**.

lpq

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 off-line. 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 off-line or having trouble (e.g., 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**.

lprm

lprm: *printer*: cannot restart printer daemon

This case is the same as when **lpr** prints that the daemon cannot be started.

lpd

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 (i.e., 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.

lpc

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 "cannot ..." are usually due to incorrect ownership and/or protection mode of the lock file, spooling directory or the **lpc** program.

General TCP/IP Error Conditions

Socket: I/O Error

This is a general indication of problems with the TCP connection. Restarting */sys/tcp/tcp_server*, 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 9

Installing and Configuring uucp

Uucp is the name of the central program in a group of programs that, together, permit communication between DOMAIN/IX systems using either dial-up or hardwired connections. The first version of the system was designed and implemented at Bell Labs. Today, it is commonly used for file transfers and remote command execution. This chapter explains how **uucp** works and includes information about installing and configuring **uucp** on your DOMAIN/IX system.

Introduction

Uucp is a batch-type operation. Files are created in a spool directory for processing by the **uucp** daemons. There are three types of files used for the execution of work.

Data files contain data for transfer to remote systems.

Work files contain directions for file transfers between systems.

Execution files contain directions for executing DOMAIN/IX commands that involve the resources of one or more systems.

The **uucp** system consists of four primary and several secondary programs. The primary programs are:

uucp This program creates work files and gathers data files in the **uucp** spool directory.

uux This program creates work files and execute files. It also gathers data files for the remote execution of DOMAIN/IX commands.

uucico This program executes the work files for data transmission.

uuxqt This program executes DOMAIN/IX commands from command files.

Note: On DOMAIN systems, a program called **uucico** runs on all nodes that have DOMAIN/IX software. Another program, called **uucico.real**, runs only on the node(s) from which the actual connection (to another system) is made.

Secondary **uucp** programs in *bsd4.2* **uucp** are:

uulog This program updates the log file with new entries and reports on the status of uucp requests.

uuclean
This program removes old files from the spool directory.

uusend This program sends a file to a given location on a remote system.

uusnap This program displays a tabular synopsis of the current **uucp** activity at your site.

In the remainder of this chapter, we describe the operation of each program, the installation of the system, the security aspects of the system, the files required for execution, and the administration of the system.

DOMAIN/IX *bsd4.2* uucp

There is an implementation peculiarity in *bsd4.2* **uucp** of which you should be aware. The *bsd4.2* version of **uucp** creates a number of subdirectories in the directory */usr/spool/uucp,* one for each type of **uucp** file. "**D.**" files are stored in */usr/spool/uucp/D.*, "**C.**" files are stored in */usr/spool/uucp/C.*, and so on. The *sys5* version of **uucp** does not make these subdirectories.

uucp — UNIX-TO-UNIX FILE COPY

The **uucp** command looks to the user much like the DOMAIN/IX command **cp**. The syntax is

uucp [*option(s)*] *source* ... *destination*

where *source* and *destination* may contain the prefix *system-name*! that indicates the system on which the file(s) can be found (or the one to which the files will be copied).

Note: C shell users should be sure to escape (through quotation or use of \) any shell metacharacters in **uucp** command lines.

Uucp interprets the following options.

-d Make directories when necessary for copying the file.

-c Don't copy source files to the spool directory, but use the specified source when the actual transfer takes place.

-g*letter* Put letter in as the grade in the name of the work file. (This can be used to change the order of work for a particular machine.)

-m Send mail on completion of the work.

The following options are used primarily for debugging:

-r Queue the job but do not start the **uucico** program.

-s*dir* Use directory *dir* for the spool directory.

-x*num* *Num* is the level of debugging output desired.

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. The source name may contain shell metacharacters. If a source argument has a *system-name*! prefix for a remote system, the filename expansion will be done on the remote system.

The command

```
uucp *.c usg!/usr/dan
```

will set up the transfer of all files whose names end with ".c" to the "/usr/dan" directory on the "usg" machine.

The source and/or destination names may also contain a *~user* prefix. This translates to the login directory on the specified system.

Note: The "home" character, normally tilde (~), can be redefined in the C Shell. In this chapter, we assume that it has not be so redefined.

For names with partial pathnames, the current directory is prepended to the file name. File names with ../ are not permitted.

The command

```
uucp usg!~dan/*.h ~dan
```

will set up a transfer of files whose names end with ".h" in dan's login directory on system "usg" to dan's local login directory.

For each source file, the program will check the source and destination file names and the system–part of each pathname to classify the work into one of five types.

[1] Copy source to destination on local system.

[2] Receive files from other systems.

[3] Send files to a remote systems.

[4] Send files from remote systems to another remote system.

[5] Receive files from remote systems when the source contains shell metacharacters.

After the work has been set up in the spool directory, **uucp** calls upon **uucico** to contact the other machine and execute the work (unless the **-r** option was specified).

Copying on the Local System

If the source and destination are both on the local system, **uucp** simply calls **cp** and copies the file from source to destination. The **-d** and the **-m** options are not honored in this case.

Receiving Files from Other Systems

If the source is on a remote system, a "work file" is created for each file requested and put in the spool directory with the following fields, each separated by a blank. (All work files and execute files use a blank as the field separator.)

- R
- The full path–name of the source or a ~user/path–name. The *~user* part will be expanded on the remote system.
- The full path–name of the destination file. If the *~user* notation is used, it will be immediately expanded to be the login directory for the user.
- The user's login name.
- A "-" followed by an option list. (Only the **-m** and **-d** options will appear in this list.)

Sending Files to a Remote System

If the destination file is on a remote system, a work file is created for each source file, then the source file is copied into a "data file" in the spool directory. (A **-c** option on the **uucp** command will prevent the data file from being made. In this case, the file will be transmitted from the indicated source.) The fields of each entry are given below.

- S
- The full-path name of the source file.
- The full-path name of the destination or ~user/file-name.
- The user's login name.
- A "-" followed by an option list.
- The name of the data file in the spool directory.
- The file mode bits of the source file in octal print format (e.g. 0666).

Transfers from one Remote System to Another

If both source and destination files are on remote systems, **uucp** generates files that are subsequently executed by the **uucico** program running on a remote machine.

uux - UNIX TO UNIX EXECUTION

The **uux** command is used to set up the execution of a DOMAIN/IX command where the execution machine and/or some of the files are remote. The syntax of **uux** is

uux [*option(s)*] *command string*

where *command string* is made up of one or more arguments. All shell metacharacters must be protected either by quoting the entire *command string* or quoting the character as a separate argument. Within the *command string*, the command and file names may contain a *system- name*! prefix. Arguments that do not contain a "!" will not be treated as files (i.e., they will not be copied to the execution machine.) The – is used to indicate that the standard input for *command–string* should be inherited from the standard input of the **uux** command. The options, mostly useful for debugging, are:

-r Don't start **uucico** or **uuxqt** after queuing the job;

-x*num* *Num* is the level of debugging output desired.

The command

```
pr abc | uux - usg!lpr
```

will set up the output of "pr abc" as standard input to an **lpr** command to be executed on system "usg".

Uux generates an "execute file" that contains the names of the files required for execution (including standard input), the user's login name, the destination of the standard output, and the command to be executed. This file is either put in the spool directory for local execution or sent to the remote system by **uucp**.

For required files that are not on the execution machine, **uux** will generate receive command files. These command–files will be put on the execution machine and executed by the **uucico** program.

(This will work only if the local system has permission to put files in the remote spool directory as controlled by the remote *USERFILE*.) The execute file will be processed by the **uuxqt** program on the execution machine. It is made up of several lines, each of which contains an identification character and one or more arguments. The order of the lines in the file is not relevant and some of the lines may not be present. Each line is described below.

User Line

This line has the form

U *user system*

where the *user* and *system* are the requester's login name and system.

Required File Line

This line has the form

F *file–name real–name*

where the *file–name* is the generated name of a file for the execute machine and *real–name* is the last part of the actual file name (contains no path information). Zero or more of these lines may be present in the execute file. The **uuxqt** program will check for the existence of all required files before the command is executed.

Standard Input Line

This line has the form

I *file–name*

The standard input is either specified by a "<" in the command–string or inherited from the standard input of the **uux** command if the "-" option is used. If a standard input is not specified, "/dev/null" is used.

Standard Output Line

This line has the form

O *file–name system–name*

The standard output is specified by a ">" within the command–string. If a standard output is not specified, "/dev/null" is used. (Note that the use of ">>" is not implemented.)

Command Line

This line has the form

C *command* [*argument(s)*]

The *arguments* are those specified in the command–string. The standard input and standard output will not appear on this line. All required files will be moved to the execution directory (a subdirectory of the spool directory) and the DOMAIN/IX command will be executed using the Shell specified in the **uucp.h** header file. In addition, the "PATH" field from the *L.cmds* file is prepended to the command line as specified in **uuxqt**.

After execution, the standard output is copied or set up to be sent to the proper place.

uucico—COPY IN, COPY OUT

The **uucico** program performs the following major functions:

- Scan the spool directory for work.
- Place a call to a remote system.
- Negotiate a line protocol to be used.
- Execute all requests from both systems.
- Log work requests and work completions.

Uucico may be started in several ways;

- by a system daemon like **cron**(1),
- by one of the **uucp**, **uux**, **uuxqt** or **uucico** programs;
- directly—usually only for testing—by the user;
- by a remote system.

When started by any of the first three methods, the program is considered to be in *MASTER* mode. In this mode, a connection will be made to a remote system. If started by a remote system, the program is considered to be in *SLAVE* mode.

The *MASTER* mode will operate in one of two ways. If the –s*sys* option is not specified, the program will scan the spool directory for systems to call. If a system name is specified, that system will be called, and work will only be done for that system.

Uucico recognizes the following options.

–r1 Start the program in *MASTER* mode. This is used when **uucico** is started by a program or **cron** shell.

–s*sys* Do work only for system *sys*. If -s is specified, a call to the specified system will be made even if there is no work for system *sys* in the spool directory. This is useful for polling systems which do not have the hardware to initiate a connection.

–d*dir* Use directory *dir* for the spool directory (primarily for debugging.)

–x*num* *Num* is the level of debugging output desired.

Uucico and uucico.real

In a distributed system, it's important to distinguish between those nodes that are able to make a connection to a remote host and those that are not. Typically, one node in a DOMAIN system will be designated the "**uucp** server." Requests for **uucp** services entered at other nodes in the network will be handled by this server node, since attempts to handle them by other nodes would invariably fail due to lack of the appropriate connect hardware.

As has been noted, DOMAIN 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 actual **uucico** program. Normally, **uucico.real** is invoked locally by **cron**, or when a remote host intiates a **uucp** connection with the DOMAIN system in slave mode. Work queued by you will be processed when **uucico.real** is invoked by either method. The only way for a user to directly invoke **uucico.real** is to run it on the node on which it is installed.

The directory */usr/spool/uucppublic/user_data* should include a file named *startup_sh* that contains the following lines.

```
/usr/lib/uucp/uucico.real
/com/tctl -line 1 -speed 1200 -bpc 8 -error noframing -noinsync -nosync
logout
```

This performs a function equivalent to specifying **uucico.real** in the "shell" field in the */etc/passwd* file for the "uucp" logins.

In the following subsections, which detail the operation of **uucico**, we are actually referring to **uucico.real**.

Scanning For Work

The names of the work related files in the spool directory have the format

type.system–name grade number

where *type* is one of the following uppercase letters.

C copy command file,
D data file,
X execute file

System–name is the remote system. *Grade* is a character. *Number* is a four digit, padded sequence number.

Note: In the bsd4.2 version, these files are kept in subdirectories */usr/spool/uucp/C.*, */usr/spool/uucp/D.*, and so on.

The file

C.res45n0031

would be a work file for a file transfer between the local machine and the "res45" machine.

The scan for work is done by looking through the spool directory for work files (prefixed with the sequence *C.*). A list is made of all systems to be called. **Uucico.real** will then call each system and process all work files.

Calling the Remote System

The call is made using information from several files which reside in the uucp program directory. At the start of the call process, a lock is set to forbid multiple conversations between the same two systems.

The system name is found in the *L.sys file*. The information contained for each system is:

[1] system name,

[2] times to call the system (days-of-week and times-of-day);

[3] device or device type to be used for call;

[4] line speed;

[5] phone number if field [3] is *ACU* or the device name (same as field [3]) if not *ACU*;

[6] login information (multiple fields);

The time field is checked against the present time to see if the call should be made.

The *phone number* may contain abbreviations (e.g., mh, py, boston) which get translated into dial sequences using the *L-dialcodes* file.

The *L-devices* file is scanned using fields [3] and [4] from the *L.sys* file to find an available device for the call. The program will try all devices which satisfy [3] and [4] until the call is made, or no more devices can be tried. If a device is successfully opened, a lock file is created so that another copy of **uucico** will not try to use it. If the call is complete, the login information (field [6] of *L.sys*) is used to login.

The conversation between the two **uucico** programs begins with a handshake started by the called — or *SLAVE* — system. The *SLAVE* sends a message to let the *MASTER* know it is ready to receive the system identification and conversation sequence number. The response from the *MASTER* is verified by the *SLAVE* and if acceptable, protocol selection begins. The *SLAVE* can also reply with a "call-back required" message in which case, the current conversation is terminated.

Line Protocol Selection

The remote system sends a message

P*proto-list*

where *proto-list* is a string of characters, each representing a line protocol.

The calling program checks the *proto-list* for a letter corresponding to an available line protocol and returns a *use-protocol* message. The *use-protocol* message is

U*code*

where *code* is either a one character protocol letter or *N* which means there is no common protocol.

Work Processing

The initial roles (*MASTER* or *SLAVE*) for the work processing determine the mode in which each program starts. (The *MASTER* has been specified by the **-r1** option of **uucico**.) The *MASTER* program does a work search similar to the one used when scanning for work.

There are five messages used during the work processing, each specified by the first character of the message. They are;

S send a file

R receive a file

C copy complete

X	execute a **uucp** command
H	hangup

The *MASTER* will send **R**, **S** or **X** messages until all work from the spool directory is complete, at which point an **H** message will be sent. The *SLAVE* will reply with **SY**, **SN**, **RY**, **RN**, **HY**, **HN**, **XY**, **XN**, corresponding to *Yes* or *No* for each request.

The send and receive replies are based on permission to access the requested file/directory using the *USERFILE* and read/write permissions of the file/directory. After each file is copied into the spool directory of the receiving system, a copy–complete message is sent by the receiver of the file. The message **CY** will be sent if the file has successfully been moved from the temporary spool file to the actual destination. Otherwise, a **CN** message is sent. (In the case of **CN**, the transferred file will be in the spool directory with a name beginning with "TM'.) The requests and results are logged on both systems.

The hangup response is determined by the *SLAVE* program by a work scan of the spool directory. If work for the remote system exists in the *SLAVE*'s spool directory, an **HN** message is sent and the programs switch roles. If no work exists, an **HY** response is sent.

Conversation Termination

When a **HY** message is received by the *MASTER*, it is echoed back to the *SLAVE* and the protocols are turned off. Each program sends a final "OO" message to the other. The original *SLAVE* program will clean up and terminate. The *MASTER* will proceed to call other systems and process work as long as possible or terminate if a –s option was specified.

uuxqt – uucp COMMAND EXECUTION

The **uuxqt** program is used to execute *execute files* generated by **uux**. The **uuxqt** program may be started by either the **uucico** or **uux** programs. The program scans the spool directory for *execute files* (prefix "X."). Each one is checked to see if all the required files are available and if so, the command line or send line is executed.

The *execute file* is described in the "**Uux**" section above.

The execution is accomplished by invoking a **sh -c** of the command line after appropriate standard input and standard output have been opened. If a standard output is specified, the program will create a send command or copy the output file as appropriate.

uulog – uucp LOG INQUIRY

For each program invocation, the **uucp** programs make entries in a master log file. The **uulog** program outputs specified log entries. The output request is specified by the use of the following options:

–s*sys*	Print entries where *sys* is the remote system name;
–u*user*	Print entries for user *user*.

The intersection of lines satisfying the two options is output. A null *sys* or *user* means all system names or users respectively.

uuclean - uucp SPOOL DIRECTORY CLEANUP

This program is typically run once a day by **cron**. Its function is to examine the spool directory and remove files that are more than 3 days old. These are usually files for work which can not be completed.

The following options are available.

-d*dir* Scan directory *dir*.

-m Send mail to the owner of each file being removed. (Note that most files put into the spool directory will be owned by the owner of the **uucp** programs since the setuid bit w be set on these programs. The mail will therefore most often go to the owner of the **uucp** programs.)

-n*hours*
Change the aging time from 72 hours to *hours* hours.

-p*pre* Examine files with prefix *pre* for deletion. (Up to 10 file prefixes may be specified.)

-x*num* This is the level of debugging output desired.

SECURITY

The **uucp** system, left unrestricted, will let any outside user execute any commands and copy in/out any file which is readable/writable by the **uucp** login user. It is up to the individual sites to be aware of this and apply the protections that they feel are necessary.

There are several security features available aside from the normal file mode protections. These must be set up by the installer of the **uucp** system.

- The login for **uucp** does not get a standard shell. Instead, the **uucico** program is started. Therefore, the only work that can be done is through **uucico.**
- A path check is done on file names that are to be sent or received. The *USERFILE* supplies the information for these checks. The *USERFILE* can also be set up to require call-back for certain login-ids. (See the "Files required for execution" section for the file description.)
- The file */usr/lib/uucp/L.cmds* is a list of commands that **uucp** considers legal for remote execution. The installer may modify this file as necessary. If the *L.cmds* file is missing, **uuxqt** uses a default list of legal commands. For the *bsd4.2* version, this list consists of **rmail, rnews,** and **uusend.**
- The *L.sys* file should be owned by **uucp** and have mode 0400 to protect the phone numbers and login information for remote sites. (Programs **uucp, uucico, uux, uuxqt** should be also owned by **uucp** and have the setuid bit set.)

INSTALLING uucp ON DOMAIN SYSTEMS

In this section, we explain how to install **uucp** on a DOMAIN system.

The Installation Script

We supply two scripts for installing DOMAIN/IX at your site. One — the "administrative install," called *install_sysadmin* — has a subsection which handles installation of **uucp**. The installation script checks for the necessary account names, and creates them if they do not exist; it also sets the access modes (and ACL's) on various filesystem objects, including **uucp** files.

Selecting a Name for the Local System

Uucp requires you to choose a name by which your DOMAIN system (**uucp** site) will be known to other **uucp** sites. Names can be any number of lowercase letters, although **uucp** will recognize the first seven characters and ignore the rest.

When the name has been selected, record it in the file */etc/net/uname,* then verify that the name can be accessed by running the command

```
% uuname -l
```

which displays the selected name.

Note: You will need to log in as either "root" or "uucp" in order to edit */etc/net/uname*.

Making Subdirectories

The *bsd4.2* version of **uucp** expects to find subdirectories of the form **D.***name* and **D.***name***X** in */usr/lib/uucp*. We provide a shell script, */usr/lib/uucp/mksubdirs.sh* in the *bsd4.2* version of */usr*. Run this script after you have put the site name in */etc/net/uname*.

REQUIRED FILES

This section details the files that **uucp** needs to access in the course of its normal operations. These files are

- */usr/lib/uucp/L.sys*
- */usr/lib/uucp/L.dialcodes*
- */usr/lib/uucp/L.devices*
- */usr/lib/uucp/L.cmds*
- */usr/lib/uucp/USERFILE*
- */etc/net/uname*

Note: The field separator for all files is a space unless otherwise specified.

We have provided example versions of these files in the appropriate places in the DOMAIN/IX distribution filesystem. Examine these files as you read the following descriptions.

L-devices

This file contains entries for the call-unit devices and hardwired connections to be used by **uucp**. The special device files are assumed to be in the */dev* directory. The format for each entry is

caller line call-unit speed dialer

where;

caller is the type of device that will be making the connection. For *bsd4.2* **uucp**, typical specifications for caller are *ACU* (for Automatic Call Unit) or *DIR* (for a Direct Connection).

line is the device for the line (e.g. sio1),

call-unit
is an unused field in DOMAIN systems.

speed is the line speed (baud rate).

dialer is the type of *ACU* used (e.g., hayes, vadic, ventel, or other type name).

The *bsd4.2* version of **uucp** supports the following modems.

- Hayes
- Vadic
- Ventel

There are sample *L-devices* files in the */usr/lib/uucp* directory.

L-dialcodes

This file contains entries with location abbreviations used in the *L.sys* file (e.g. py, mh, boston). The entry format is

abb dial-seq

where;

abb is the abbreviation,

dial-seq is the dial sequence to call that location.

There are sample *L–dialcodes* files in the */usr/lib/uucp* directory for each version of DOMAIN/IX.

uname

This is the file (*/etc/net/uname*) where the system name resides. While the login name used by a remote host should not be the same as the login name of a local user, several remote computers may employ the same login name. Each uucp site is given a unique system name that is transmitted at the start of each call. This name identifies the calling machine to the called machine.

USERFILE

This file contains user accessibility information. It specifies which files can be accessed by a normal user of the local machine, which files can be accessed from a remote computer, which login name is used by a particular remote computer, and whether a remote computer should be called back in order to confirm its identity. Each line in *USERFILE* has the following format

> *login sys* [c] *pathname ...*

where;

login is the login name for a user or the remote computer,
sys is the system name for a remote computer,
c is the optional *call–back required* flag,
pathname is a pathname prefix that is acceptable for *user*.

The constraints are implemented as follows.

[1] When the program is obeying a command stored on the local machine (*MASTER* mode), the pathnames allowed are those given for the first line in the *USERFILE* that has a login name that matches the login name of the user who entered the command. If no such line is found, the first line with a null login name is used.

[2] When the program is responding to a command from a remote machine (*SLAVE* mode), the pathnames allowed are those given for the first line in the file that has the system name that matches the system name of the remote machine. If no such line is found, the first one with a null system name is used.

[3] When a remote computer logs in, the login name that it uses must appear in the *USERFILE*. There may be several lines with the same login name but one of them must either have the name of the remote system or must contain a null system name.

[4] If the line matched in ([3]) contains a "c", the remote machine is called back before any transactions take place.

The line

> *u,m /usr/xyz*

allows machine *m* to login with name *u* and request the transfer of files whose names start with "*/usr/xyz*".

The line

> *dan, /usr/dan*

allows the ordinary user *dan* to issue commands for files whose name starts with "*/usr/dan*".

The lines

```
u,m /usr/xyz /usr/spool
u, /usr/spool
```

allows any remote machine to login with name *u*, but if its system name is not *m*, it can only ask to transfer files whose names start with "*/usr/spool*".

The lines

```
root, /
, /usr
```

allow any user to transfer files beginning with "/usr" but the user with login *root* can transfer any file.

L.sys

Each entry in this file represents a system that your **uucp** site can call. The fields are described below.

sysname
The name of the remote system.

time A string which indicates the days-of-week and times-of-day when the system should be called (e.g., MoTuTh0800-1730). Times may be stated using the keywords *Su Mo Tu We Th Fr Sa* for the seven days of the week, *Wk* for "any week-day," and *Any* for "any day." The time should be a range of times (e.g., 0800-1230). If no *time* is specified, **uucp** assumes that it may call at any time of day.

device This is either *ACU* (Automatic Call Unit) or *DIR* (DIRect connection). For the hardwired case, the last part of the special file name is used (e.g., sio1).

speed This is the line speed for the call (e.g., 300).

phone The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation is one which appears in the *L-dialcodes* file (e.g., mh5900, boston995-9980). For the hardwired devices, this field contains the same string as used for the device field.

login The login information is given as a series of fields and subfields in the format

```
expect  send expect  send ...
```

where *expect* is the string expected to be read and *send* is the string to be 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.

Two special names can be sent during the login sequence. The string *EOT* will send an ASCII EOT character and the string *BREAK* will try to send an ASCII BREAK character. (**Uucp** simulates the BREAK character by using line-speed changes and null characters. This may not work on all devices and/or systems.) A typical entry in the *L.sys* file would be

```
sys Any ACU 300 mh7654 login uucp ssword: word
```

The *expect* algorithm looks at the last part of the string as illustrated in the password field.

There are sample *L.sys* files in the */usr/lib/uucp* directory for each version of DOMAIN/IX.

ADMINISTRATION

This section explains the responsibilities of the **uucp** system administrator. Some administration can be accomplished by shell scripts that are initiated by **crontab** entries. Others will require manual intervention. Some sample shell scripts are included in the */usr/lib/uucp* directory for each version of DOMAIN/IX.

TM - Temporary Data Files

These files are created in the spool directory while files are being copied from a remote machine. Their names have the form

```
TM.pid.ddd
```

where *pid* is a process–id and *ddd* is a sequential three digit number starting at zero for each invocation of **uucico** and incremented for each file received.

After the entire remote file is received, the *TM* file is moved/copied to the requested destination. If processing is abnormally terminated or the move/copy fails, the file will remain in the spool directory.

The leftover files should be periodically removed; **uuclean** is useful in this regard. The following command line is from a *bsd4.2* **uucp** cleanup script.

```
% uuclean –pTM –d/usr/spool/uucp/TM
```

It removes all **TM** files that are more than three days old.

System Status Files

These files are created in the spool directory by the **uucico** program. They contain information about failures in the areas of login, dialup or sequence check and will contain a *TALKING* status when two machines are conversing. The form of the file name is

```
ST.sys
```

where *sys* is the remote system name.

For ordinary failures (dialup, login), the file will prevent repeated tries for about one hour. For sequence check failures, the file must be removed before any future attempts to converse with that remote system.

If the file is left due to an aborted run, it may contain a *TALKING* status. In this case, the file must be removed before a conversation is attempted.

LCK - Lock Files

Lock files are created for each device in use (e.g., automatic calling unit) and each system conversing. This prevents duplicate conversations and multiple attempts to use the same devices. The form of the lock file name is

LCK.*str*

where *str* is either a device or system name. The files may be left in the spool directory if runs abort. They will be ignored (reused) after a time of about 24 hours. When runs abort and calls are desired before the time limit, the lock files should be removed.

Shell Scripts

The **uucp** program will spool work and attempt to start the **uucico** program, but the starting of **uucico** will sometimes fail (no devices available, login failures etc.). Therefore, the **uucico** program should be periodically started. The command to start **uucico** can be put in a shell script and started by a **crontab** entry on an hourly basis. Here's a sample file for the *bsd4.2* environment.

```
: bsd4.2 Hourly uucp script
 cd /usr/lib/uucp
: poll of sites you want to connect to hourly
 uucico.real -r1 -ssys1
 uucico.real -r1 -ssys2
: attempt to ship remaining files
 uucico.real -r1
```

Note that the "**-r1**" option is required to start the **uucico** program in *MASTER* mode.

You can write another shell script that runs daily and removes **TM**, **ST** and **LCK** files as well as **C.** or **D.** files for work that could not be accomplished (e.g. bad phone number, login change, etc.). Here's an example. This one is available in the file */bsd4.2/usr/lib/uucp/uu.daily*.

```
: bsd4.2 daily uucp script
: assumes you already have subdirectories.
cd /usr/lib/uucp
uuclean -pLTMP. -n24
uuclean -d/usr/spool/uucp/TM. -pTM. -n240
uuclean -d/usr/spool/uucp/X. -pX. -n240
uuclean -d/usr/spool/uucp/C. -pC. -n240
uuclean -d/usr/spool/uucp/D. -pD. -n240
uuclean -d/usr/spool/uucp/D.‘uuname -l‘ -pD. -n240
uuclean -d/usr/spool/uucp/D.‘uuname -l‘X -pD. -n240

 SPOOL=/usr/spool/uucp
mv $SPOOL/LOGFILE $SPOOL/LOGFILE.old
```

This script is designed for use in the *bsd4.2* environment, since it looks in the subdirectories **D.***'uuname'* and **D.***'uuname'X*. Note that we use the "**-n24**" option to clean up **LTMP** files more than 24 hours old, and the "**-n240**" option on everything else. The absence of the "**-n**" option will use a three-day time limit.

A daily or weekly shell script should also be created to remove or save old *LOGFILE*s.

Added Information since SR9.0

When the "device:" field of L.sys specifies "DIR" (for a direct connection), the *L-devices* file should specify which SIO line to use (e.g., "sio1" for connections made via /dev/sio1). As an example, if your site has a direct **uucp** connection at 9600 baud between SIO line 2 of a DOMAIN node and a foreign host, *L.sys* should include a line of the form:

```
hostname Any DIR 9600 sio2 in:--in: uucp word: uucp_password
```

and L-devices should include the following line:

```
DIR sio2 unused 9600 direct
```

In addition to modifying those files **uucp** requires to support connections to other machines, you also need to create or modify various files in the */sys/node_data* directory of the node which will be the incoming **uucp** server at your site (the one to which the incoming connection -- hardwired or dial-up -- is made). Any node to which an incoming SIO connection is to be made must be running **siomonit** as a server process. **Siomonit** invokes the **siologin** process whenever someone attempts to connect via the SIO line. To ensure that the **siomonit** process is restarted after the server node has crashed or been shut down, place the following line in that node's *'node_data/startup.[type]* file. The example below is from a DN460 node used to receive incoming **uucp** connections via SIO line 1.

```
# from `node_data/startup.191
# uucp incoming (listener) node only
#
# start the siomonit process to watch the sio line
cps /sys/siologin/siomonit `node_data/siomonit_file -n siomonit
```

Then create the file *'node_data/siomonit_file* and place this line in it (again, the example is for a connection to SIO line 1):

```
-repeat /dev/sio1 -dialin -n sio1 /com/sh -startup
```

Siologin runs the **/com/sh** script *'node_data/siologin/startup_sio.sh* every time it starts up, passing it the number of the SIO line as the first argument. This script resets the characteristics of SIO lines 1 and 2 to certain defaults. Most **uucp** sites will require some editing of this file, as shown in the example below.

```
#! /com/sh
# This file is run as a shell script when siologin starts a new login
# process on an sio line.  The line number is provided as the first
# argument.
# Force-unlock the line.
#
ulkob /dev/sio^1 -f
# Set the speed and other line-dependent settings.
#
if eqs ^1 1 then
# default version commented out
#  tctl -line ^1 -default -speed 1200 -dcd_enable -insync -nosync
# uucp incoming dialup server should use this line instead
    tctl -line ^1 -speed 1200 -bpc 8 -nosync -noinsync
endif
if eqs ^1 2 then
#  default version commented out
#  tctl -line ^1 -default -speed 9600 -dcd_enable -insync -nosync
#  uucp incoming direct connect server should use this line instead
  tctl -line ^1 -speed 9600 -bpc 8 -nosync -noinsync
endif
```

This example sets up line 1 to handle incoming **uucp** dial-up at 1200 baud and line 2 to handle incoming **uucp** direct connections at 9600 baud. Be sure to add the #! /com/sh line at the top. You must also make sure the file */usr/spool/uucppublic/user_data/sh/startup* exists, and that it contains the lines:

```
# /usr/spool/uucppublic/user_data/sh/startup
/usr/lib/uucp/uucico.real
logout
```

The installation procedure sets HOME for user "uucp" to */usr/spool/uucppublic.*

Chapter 10

Sendmail Configuration and Usage

Routing mail through a heterogenous internet presents many new 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.

Sendmail 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 (NCP/RFC733) Arpanet, new (TCP/RFC822) Arpanet, UUCP, and Phonenet. Sendmail also implements an SMTP server, message queueing, and aliasing.

Note: This chapter is largely taken from the original *Sendmail Installation and Operation Guide*, and the paper *Introduction to Sendmail*, both by Eric Allman.

Introduction

Sendmail 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-username pair. Host names and numbers have to be administered by a central authority, but usernames can be assigned locally to each host.

In an internet, multiple networks with different characterstics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled trivially 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 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.

Internet standards seek to eliminate these problems. Initially, these proposed expanding the address pairs to address triples, consisting of {network, host, resource} triples. Network numbers must be universally agreed upon, and hosts can be assigned locally on each network. The user–level presentation was quickly expanded to address domains, comprised of a local resource identification and a hierarchical domain specification with a common static root. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form "eric@a.cc.berkeley.arpa" describes only the logical organization of the address space.

Sendmail 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 old arbitrary address syntaxes, resolving ambiguities using heuristics specified by the system administrator, as well as domain–based addressing. 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.

Section 1 discusses the design goals for *sendmail.* Section 2 gives an overview of the basic functions of the system. In section 3, details of usage are discussed. Section 4 compares *sendmail* to other internet mail routers, and an evaluation of *sendmail* is given in section 5, including future plans.

Design Goals

Design goals for **sendmail** include:

1. Compatibility with the existing mail programs, including Bell version 6 mail, Bell version 7 mail [UNIX83], Berkeley *Mail* [Shoens79], BerkNet mail [Schmidt79], and UUCP mail [Nowitz78a, Nowitz78b]. ARPANET mail [Crocker77a, Postel77] was also required. (Please note that citations of the form [NameYear] refer to the list of papers at the end of this chapter.)

2. Reliability, in the sense of guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost. This goal was considered essential because of the emphasis on mail in our environment. It has turned out to be one of the hardest goals to satisfy, especially in the face of the many anomalous message formats produced by various ARPANET sites. For example, certain sites generate improperly formated addresses, occasionally causing error–message loops. Some hosts use blanks in names, causing problems with UNIX mail programs that assume that an address is one word. The semantics of some fields are interpreted slightly differently by different sites. In summary, the obscure features of the ARPANET mail protocol really *are* used and are difficult to support, but must be supported.

3. Existing software to do actual delivery should be used whenever possible. This goal derives as much from political and practical considerations as technical.

4. Easy expansion to fairly complex environments, including multiple connections to a single network type (such as with multiple UUCP or Ethernets [Metcalfe76]). This goal requires consideration of the contents of an address as well as its syntax in order to determine which gateway to use. For example, the ARPANET is bringing up the TCP protocol to replace the old NCP protocol. No host at Berkeley runs both TCP and NCP, so it is necessary to look at the ARPANET host name to determine whether to route mail to an NCP gateway or a TCP gateway.

5. Configuration should not be compiled into the code. A single compiled program should be able to run as is at any site (barring such basic changes as the CPU type or the operating system). We have found this seemingly unimportant goal to be critical in real life. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites like to "fiddle" with anything that they will be recompiling anyway.

6. **Sendmail** must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.

7. Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users who are using specialized mailers that use a different format to build their environment without changing the system, and facilitates specialized functions (such as returning an "I am on vacation" message).

Network traffic should be minimized by batching addresses to a single host where possible, without assistance from the user.

Overview

System Organization

Sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user interface program (UIP) such as Berkeley *Mail*, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission, except when mailing to a file, when **sendmail** does the delivery directly. This discipline allows the insertion of new mailers at minimum cost. In this sense, **sendmail** resembles the Message Processing Module (MPM) of [Postel79b].

Interfaces to the Outside World

There are three ways **sendmail** can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX argument vector/return status, speaking SMTP over a pair of UNIX pipes, and speaking SMTP over an interprocess(or) channel.

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.

SMTP over pipes

The SMTP protocol [Postel82] 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.

SMTP over an IPC connection

This technique is similar to the previous technique, except that it uses a 4.2BSD IPC channel [UNIX83]. 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.

Operational Description

When a sender wants to send a message, it issues a request to **sendmail** using one of the three methods described above. **Sendmail** operates in two distinct phases. In the first phase, it collects and stores the message. In the second phase, message delivery occurs. If there were errors during processing during the second phase, **sendmail** creates and returns a new message describing the error and/or returns an status code telling what went wrong.

Argument processing and address parsing

If **sendmail** is called using one of the two subprocess techniques, the arguments are first scanned and option specifications are processed. Recipient addresses are then collected, either from the command line or from the SMTP RCPT command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the addresses is done at this step: syntax is checked, and local addresses are verified, but detailed checking of host names and addresses is deferred until delivery. Forwarding is also performed as the local addresses are verified.

Sendmail appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages deliverd to the same recipient, as might occur if a person is in two groups.

Message collection

Sendmail then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message except that they must be lines of text (i.e., binary data is not allowed). The header is parsed and stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no addresses were valid. The message will be returned with an error.

Message delivery

For each unique mailer and host in the recipient list, **sendmail** calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to **sendmail**. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.

Queueing for retransmission

If the mailer returned an status that indicated that it might be able to handle the mail later, **sendmail** will queue the mail and try again later.

Return to sender

If errors occur during processing, **sendmail** returns the message to the sender for retransmission. The letter can be mailed back or written in the file "dead.letter" in the sender's home directory. Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only affect the error message -- the "return to sender" function is always handled in one of these two ways.

Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a "From:" line and a "Full-name:" line can be merged under certain circumstances.

Configuration File

Almost all configuration information is read at runtime from an ASCII file, encoding macro definitions (defining the value of macros used internally), header declarations (telling **sendmail** the format of header lines that it will process specially, i.e., lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which is used to parse and rewrite the addresses).

To improve performance when reading the configuration file, a memory image can be provided. This provides a "compiled" form of the configuration file.

Usage and Implementation

Arguments

Arguments may be flags and addresses. Flags set various processing options. Following flag arguments, address arguments may be given, unless we are running in SMTP mode. Addresses follow the syntax in RFC822 [Crocker82] for ARPANET address formats. In brief, the format is:

Anything in parentheses is thrown away (as a comment).

Anything in angle brackets ("<>") is preferred over anything else. This rule implements the ARPANET standard that addresses of the form

    user name <machine–address>

will send to the electronic "machine–address" rather than the human "user name."

Double quotes (") quote phrases; backslashes quote characters. Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently -- for example, *user* and *"user"* are equivalent, but *\user* is different from either of them.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control remaining parsing, although some special processing is done after rewriting local names; see below.

Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages (such as the Berkeley *msgs* program, or the MARS system [Sattley78]).

Any address passing through the initial parsing algorithm as a local address (i.e, not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar ("|") the rest of the address is processed as a shell command. If the user name begins with a slash mark ("/") the name is used as a file name, instead of a login name.

Files that have setuid or setgid bits set but no execute bits set have those bits honored if **sendmail** is running as root.

Aliasing, Forwarding, Inclusion

Sendmail reroutes mail three ways. Aliasing applies system wide. Forwarding allows each user to reroute incoming mail destined for that account. Inclusion directs **sendmail** to read a file for a list of addresses, and is normally used in conjunction with aliasing.

Aliasing

Aliasing maps names to address lists using a system–wide file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this guarantees a unique key (since there are no nicknames for the local host).

Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a ".forward" file in their home directory. If it exists, the message is *not* sent to that user, but rather to the list of users in that file. Often this list will contain only one address, and the feature will be used for network mail forwarding.

Forwarding also permits a user to specify a private incoming mailer. For example, forwarding to:

```
"|/usr/local/newmail myname"
```

will use a different incoming mailer.

Inclusion

Inclusion is specified in RFC 733 [Crocker77a] syntax:

```
:Include: pathname
```

An address of this form reads the file specified by *pathname* and sends to all users listed in that file.

The intent is *not* to support direct use of this feature, but rather to use this as a subset of aliasing. For example, an alias of the form:

```
project: :include:/usr/project/userlist
```

is a method of letting a project maintain a mailing list without interaction with the system administration, even if the alias file is protected.

It is not necessary to rebuild the index on the alias database when a :include: list is changed.

Message Collection

Once all recipient addresses are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form

```
field–name: field–value
```

Field–value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

Message Delivery

The send queue is ordered by receiving host before transmission to implement message batching. Each address is marked as it is sent so rescanning the list is safe. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list.

After a connection is established, **sendmail** makes the per–mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return–to–sender function after all delivery completes.

Queued Messages

If the mailer returns a "temporary failure" exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other salient parameter of the message. The header of the message is stored in the control file, so that the associated data file in the queue is just the temporary file that was originally collected.

Configuration

Configuration is controlled primarily by a configuration file read at startup. **Sendmail** should not need to be recomplied except

1. To change operating systems (V6, V7/32V, 4BSD).

2. To remove or insert the DBM (UNIX database) library.

3. To change ARPANET reply codes.

4. To add headers fields requiring special processing.

5.. Adding mailers or changing parsing (i.e., rewriting) or routing information does not require recompilation.

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.

DOMAIN/IX contains three sample configuration files. These files are extensively commented and may work as is for many sites.

- */bsd4.2/usr/lib/uucpproto.cf*
- */bsd4.2/usr/lib/arpaproto.cf*
- */bsd4.2/usr/lib/sendmail.cf*

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); e.g., 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 heart of address parsing in **sendmail** is a set of rewriting rules. These are an ordered list of pattern–replacement rules, (somewhat like a production system, except that order is critical), which are applied to each address. The address is rewritten textually until it is either rewritten into a special canonical form (i.e., a (mailer, host, user) 3–tuple, such as {arpanet, usc–isif, postel} representing the address "postel@usc–isif"), or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of addresses into different formats. For example, an address of the form:

    ucsfcgl!tef

might be mapped into:

    tef@ucsfcgl.UUCP

to conform to the domain syntax. Translations can also be done in the other direction.

Option setting

There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.

Comparison with other Mailers

Delivermail

Sendmail is an outgrowth of **delivermail**. The primary differences are:

Configuration information is not compiled in. This change simplifies many of the problems of moving to other machines. It also allows easy debugging of new mailers.

Address parsing is more flexible. For example, **delivermail** only supported one gateway to any network, whereas **sendmail** can be sensitive to host names and reroute to different gateways.

Forwarding and :include: features eliminate the requirement that the system alias file be writable by any user (or that an update program be written, or that the system administration make all changes).

Sendmail supports message batching across networks when a message is being sent to multiple recipients.

A mail queue is provided in **sendmail**. Mail that cannot be delivered immediately but can potentially be delivered later is stored in this queue for a later retry. The queue also provides a buffer against system crashes; after the message has been collected it may be reliably redelivered even if the system crashes during the initial delivery.

Sendmail uses the networking support provided by 4.2BSD to provide a direct interface networks such as the ARPANET and/or Ethernet using SMTP (the Simple Mail Transfer Protocol) over a TCP/IP connection.

MMDF

MMDF [Crocker79] spans a wider problem set than **sendmail**. For example, the domain of MMDF includes a "phone network" mailer, whereas **sendmail** calls on preexisting mailers in most cases.

MMDF and **sendmail** both support aliasing, customized mailers, message batching, automatic forwarding to gateways, queueing, and retransmission. MMDF supports two–stage timeout, which **sendmail** does not support.

The configuration for MMDF is compiled into the code. Dynamic configuration tables are currently being considered for MMDF, allowing the installer to select either compiled or dynamic tables.Since MMDF does not consider backwards compatibility as a design goal, the address parsing is simpler but much less flexible.

It is somewhat harder to integrate a new channel. The MMDF equivalent of a **sendmail** "mailer." into MMDF. In particular, MMDF must know the location and format of host tables for all channels, and the channel must speak a special protocol. This allows MMDF to do additional verification (such as verifying host names) at submission time.

MMDF strictly separates the submission and delivery phases. Although **sendmail** has the concept of each of these stages, they are integrated into one program, whereas in MMDF they are split into two programs.

Message Processing Module

The Message Processing Module (MPM) discussed by Postel [Postel79b] matches **sendmail** closely in terms of its basic architecture. However, like MMDF, the MPM includes the network interface software as part of its domain.

MPM also postulates a duplex channel to the receiver, as does MMDF, thus allowing simpler handling of errors by the mailer than is possible in **sendmail**. When a message queued by **sendmail** is sent, any errors must be returned to the sender by the mailer itself. Both MPM and MMDF mailers can return an immediate error response, and a single error processor can create an appropriate response.

MPM prefers passing the message as a structured object, with type–length–value tuples. This is similar to the NBS standard. Such a convention requires a much higher degree of cooperation between mailers than is required by **sendmail**. MPM also assumes a universally agreed upon internet name space (with each address in the form of a net–host–user tuple), which **sendmail** does not.

Evaluations and Future Plans

Sendmail is designed to work in a nonhomogeneous environment. Every attempt is made to avoid imposing unnecessary constraints on the underlying mailers. This goal has driven much of the design. One of the major problems has been the lack of a uniform address space, as postulated in [Postel79a] and [Postel79b].

A nonuniform address space implies that a path will be specified in all addresses, either explicitly (as part of the address) or implicitly (as with implied forwarding to gateways). This restriction has the unpleasant effect of making replying to messages exceedingly difficult, since there is no one "address" for any person, but only a way to get there from wherever you are.

Interfacing to mail programs that were not initially intended to be applied in an internet environment has been amazingly successful, and has reduced the job to a manageable task.

Sendmail has knowledge of a few difficult environments built in. It generates ARPANET FTP/SMTP compatible error messages (prepended with three–digit numbers [Neigus73, Postel74, Postel82]) as necessary, optionally generates UNIX–style "From" lines on the front of messages for some mailers, and knows how to parse the same lines on input. Also, error handling has an option customized for BerkNet.

The decision to avoid doing any type of delivery where possible (even, or perhaps especially, local delivery) has turned out to be a good idea. Even with local delivery, there are issues of the location of the mailbox, the format of the mailbox, the locking protocol used, etc., that are best decided by other programs. One surprisingly major annoyance in many internet mailers is that the location and format of local mail is built in. The feeling seems to be that local mail is so common that it should be efficient. This feeling is not born out by our experience; on the contrary, the location and format of mailboxes seems to vary widely from system to system.

The ability to automatically generate a response to incoming mail (by forwarding mail to a program) seems useful ("I am on vacation until late August....") but can create problems such as forwarding loops (two people on vacation whose programs send notes back and forth, for instance) if these programs are not well written. A program could be written to do standard tasks correctly, but this would solve the general case.

It might be desirable to implement some form of load limiting. I am unaware of any mail system that addresses this problem, nor am I aware of any reasonable solution at this time.

The configuration file is currently practically inscrutable; considerable convenience could be realized with a higher-level format.

It seems clear that common protocols will be changing soon to accommodate changing requirements and environments. These changes will include modifications to the message header (e.g., [NBS80]) or to the body of the message itself (such as for multimedia messages [Postel80]). Experience indicates that these changes should be relatively trivial to integrate into the existing system.

In tightly coupled environments, it would be nice to have a name server such as Grapvine [Birrell82] integrated into the mail system. This would allow a site such as "Berkeley" to appear as a single host, rather than as a collection of hosts, and would allow people to move transparently among machines without having to change their addresses. Such a facility would require an automatically updated database and some method of resolving conflicts. Ideally this would be effective even without all hosts being under a single management. However, it is not clear whether this feature should be integrated into the aliasing facility or should be considered a "value added" feature outside **sendmail** itself.

As a more interesting case, the CSNET name server [Solomon81] provides an facility that goes beyond a single tightly-coupled environment. Such a facility would normally exist outside of **sendmail** however.

BASIC 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. Most other configurations can be built by adjusting an existing configuration files incrementally.

Although **sendmail** is intended to run without the need for monitoring, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic **sendmail** installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install **sendmail** and keep it happy. Section three describes some parameters that may be safely tweaked. Section four has information regarding the command line arguments. Section five contains the nitty-gritty information about the configuration file. This section is for masochists and people who must write their own configuration file. The addenda give a brief but detailed explanation of a number of features not described in the rest of the paper.

There are two basic steps to installing **sendmail**. The hard part is to build the configuration table. This is a file that **sendmail** reads when it starts up that describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration table is quite complex, a configuration can usually be built by adjusting an existing off-the-shelf configuration. The second part is actually doing the installation, i.e., creating the necessary files, etc.

The remainder of this section will describe the installation of **sendmail** assuming 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.

Off-The-Shelf Configurations

The configuration files are all in the subdirectory *cf* of the *sendmail* directory. The ones used at Berkeley are in **m4**(1) format; files with names ending ".m4" are **m4** include files, while files with names ending ".mc" are the master files. Files with names ending ".cf" are the **m4** processed versions of the corresponding ".mc" file.

Two off the shelf configuration files are supplied to handle the basic cases: *cf/arpaproto.cf* for Arpanet (TCP) sites and *cf/uucpproto.cf* for UUCP sites. These are not in **m4** format. The file you need should be copied to a file with the same name as your system, e.g.,

```
cp uucpproto.cf ucsfcgl.cf
```

This file is now ready for installation as */usr/lib/sendmail.cf.*

Installation Using the Makefile

A makefile exists in the root of the **sendmail** directory that will do all of these steps for a 4.2BSD system. It may have to be slightly tailored for use on other systems.

Before using this makefile, you should already have created your configuration file and left it in the file "cf/*system*.cf" where *system* is the name of your system (i.e., what is returned by **hostname**(1)). If you do not have **hostname** you can use the declaration "HOST=*system*" on the **make**(1) command line. You should also examine the file *md/config.m4* and change the m4 macros there to reflect any libraries and compilation flags you may need.

The basic installation procedure is to type:

```
make
make install
```

in the root directory of the **sendmail** distribution. This will make all binaries and install them in the standard places. The second **make** command must be executed as the superuser (root).

Installation by Hand

Along with building a configuration file, you will have to install the **sendmail** startup into your UNIX system. If you are doing this installation in conjunction with a regular Berkeley UNIX install, these steps will already be complete. Many of these steps will have to be executed as the superuser (root).

lib/libsys.a

The library in *lib/libsys.a* contains some routines that should in some sense be part of the system library. These are the system logging routines and the new directory access routines (if required). If you are not running the new 4.2BSD directory code and do not have the compatibility routines installed in your system library, you should execute the commands:

```
cd lib
make ndir
```

This will compile and install the 4.2 compatibility routines in the library. You should then type:

```
cd lib   # if required
make
```

This will recompile and fill the library.

/usr/lib/sendmail

The binary for **sendmail** is located in */usr/lib*. There is a version available in the source directory that is probably inadequate for your system. You should plan on recompiling and installing the entire system:

```
cd src
rm -f *.o
make
cp sendmail /usr/lib
```

/usr/lib/sendmail.cf

The configuration file that you created earlier should be installed in */usr/lib/sendmail.cf*:

```
cp cf/system.cf /usr/lib/sendmail.cf
```

/usr/ucb/newaliases

If you are running **delivermail**, it is critical that the *newaliases* command be replaced. This can just be a link to **sendmail**:

```
rm -f /usr/ucb/newaliases
ln /usr/lib/sendmail /usr/ucb/newaliases
```

/usr/lib/sendmail.cf

The configuration file must be installed in */usr/lib*. This is described above.

/usr/spool/mqueue

The directory */usr/spool/mqueue* should be created to hold the mail queue. This directory should be mode 777 unless **sendmail** is run setuid, when *mqueue* should be owned by the **sendmail** owner and mode 755.

/usr/lib/aliases*

The system aliases are held in three files. The file "*/usr/lib/aliases*" is the master copy. A sample is given in "*lib/aliases*" which includes some aliases which *must* be defined:

```
cp lib/aliases /usr/lib/aliases
```

You should extend this file with any aliases that are apropos to your system.

Normally **sendmail** looks at a version of these files maintained by the **dbm**(3) routines. These are stored in "*/usr/lib/aliases.dir*" and "*/usr/lib/aliases.pag.*" These can initially be created as empty files, but they will have to be initialized promptly. These should be mode 666 if you are running a reasonably relaxed system:

```
cp /dev/null /usr/lib/aliases.dir
cp /dev/null /usr/lib/aliases.pag
chmod 666 /usr/lib/aliases.*
newaliases
```

/usr/lib/sendmail.fc

(We include this section for compatibility only; this capability is not currently supported under DOMAIN/IX.) If you intend to install the frozen version of the configuration file (for quick startup) you should create the file */usr/lib/sendmail.fc* and initialize it. This step may be safely skipped.

```
cp /dev/null /usr/lib/sendmail.fc
/usr/lib/sendmail -bz
```

/etc/rc

It will be necessary to start up the sendmail daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to insure that mail gets delivered when hosts come up.

Add the following lines to "/etc/rc" (or "/etc/rc.local" as appropriate) in the area where it is starting up the daemons:

```
if [ -f /usr/lib/sendmail ]; then
        (cd /usr/spool/mqueue; rm -f [lnx]f*)
        /usr/lib/sendmail -bd -q30m &
        echo -n ' sendmail' >/dev/console
fi
```

The "**cd**" and "**rm**" commands insure that all lock files have been removed; extraneous lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes **sendmail** has two flags: "-bd" causes it to listen on the SMTP port, and "-q30m" causes it to run the queue every half hour.

If you are not running a version of UNIX that supports Berkeley TCP/IP, do not include the **-bd** flag.

/usr/lib/sendmail.hf

This is the help file used by the SMTP HELP command. It should be copied from "*lib/sendmail.hf*":

```
cp lib/sendmail.hf /usr/lib
```

/usr/lib/sendmail.st

If you wish to collect statistics about your mail traffic, you should create the file "*/usr/lib/sendmail.st*":

```
cp /dev/null /usr/lib/sendmail.st
chmod 666 /usr/lib/sendmail.st
```

This file does not grow. It is printed with the program "*aux/mailstats.*"

/etc/syslog

You may want to run the *syslog* program (to collect log information about **sendmail**). This program normally resides in */etc/syslog*, with support files */etc/syslog.conf* and */etc/syslog.pid*. The program is located in the *aux* subdirectory of the **sendmail** distribution. The file */etc/syslog.conf* describes the file(s) that **sendmail** will log in. For a complete description of syslog, see the manual page for **syslog**(8).

/usr/ucb/newaliases

If **sendmail** is invoked as "newaliases," it will simulate the **-bi** flag (i.e., will rebuild the alias database; see below). This should be a link to */usr/lib/sendmail.*

/usr/ucb/mailq

If **sendmail** is invoked as "mailq," it will simulate the **-bp** flag (i.e., **sendmail** will print the contents of the mail queue; see below). This should be a link to */usr/lib/sendmail.*

Normal Operation

Quick Configuration Startup

(We include this section for compatibility only; this capability is not currently supported under DOMAIN/IX.) A fast version of the configuration file may be set up by using the **-bz** flag:

```
/usr/lib/sendmail -bz
```

This creates the file */usr/lib/sendmail.fc* ("frozen configuration"). This file is an image of **sendmail's** data space after reading in the configuration file. If this file exists, it is used instead of */usr/lib/sendmail.cf. Sendmail.fc* must be rebuilt manually every time *sendmail.cf* is changed.

The frozen configuration file will be ignored if a **-C** flag is specified or if **sendmail** detects that it is out of date. However, the heuristics are not strong so this should not be trusted.

The System Log

The system log is supported by the **syslog**(8) program.

Each line in the system log consists of a timestamp, 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 and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful;" log levels above ten are usually for debugging purposes.

The Mail Queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although **sendmail** ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

Printing the queue

The contents of the queue can be printed using the *mailq* command (or by specifying the **-bp** flag to **sendmail**):

```
mailq
```

This will produce a listing of the queue id's, 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 (it will not even time out!).

n This file is created when an id is being created. It is a separate file to insure 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 may only be one of these lines.

H A header definition. There may 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 may 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

Sendmail should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, **sendmail** first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without violating the protocol.

In some cases, you may find that a major host going down for a couple of days may 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. If you have a tendency toward voyeurism, you can use the **-v** flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rmdir /usr/spool/omqueue
```

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 may be aliased; e.g.,

```
eric@mit-xx: eric@berkeley
```

will not have the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp 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. This technique is used to improve performance.

Rebuilding the alias database

The DBM version of the database may 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 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:

The DBM version of the database is mode 666. -or-

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 DBM version 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 dies (due to being killed or a system crash) before completing the rebuild.

Sendmail 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 insure 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: 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 maintanence 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.

Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may 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.

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 builtins 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.

Arguments

The complete list of arguments to **sendmail** is described in detail in Addendum A. Some important arguments are described here.

Queue Interval

The amount of time between forking a process to run through the queue is defined by the **-q** flag. If you run in mode **f** or **a** this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in **q** mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your */etc/rc* file using the **-bd** flag. The **-bd** flag and the **-q** flag may be combined in one call:

```
/usr/lib/sendmail -bd -q30m
```

Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the **-q** flag (with no value). It is entertaining to use the **-v** flag (verbose) when this is done to watch what happens:

```
/usr/lib/sendmail -q -v
```

Debugging

There are a fairly large number of debug flags built into **sendmail**. Each debug flag has a number and a level, where higher levels means to print out more information. The convention is that levels greater than nine are "absurd," i.e., they print out so much information that you wouldn't normally want to see them except for debugging that particular piece of code. Debug flags are set using the **-d** option; the syntax is:

```
debug–flag:  -d debug–list
debug–list:  debug–option [, debug–option ]
debug–option: debug–range [ . debug–level ]
debug–range: integer | integer - integer
debug–level: integer
```

where spaces are for reading ease only. For example,

-d12	Set flag 12 to level 1
-d12.3	Set flag 12 to level 3
-d3–17	Set flags 3 through 17 to level 1
-d3–17.4	Set flags 3 through 17 to level 4

For a complete list of the available debug flags you will have to look at the code (they are too dynamic to keep this documentation up to date).

Trying a Different Configuration File

An alternative configuration file can be specified using the -C flag; for example,

```
/usr/lib/sendmail -Ctest.cf
```

uses the configuration file *test.cf* instead of the default */usr/lib/sendmail.cf.* If the -C flag has no value it defaults to *sendmail.cf* in the current directory.

Changing the Values of Options

Options can be overridden using the **-o** flag. For example,

```
/usr/lib/sendmail -oT2m
```

sets the **T** (timeout) option to two minutes for this run only.

Tuning

There are a number of configuration parameters you may 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).

Timeouts

All time intervals are set using a scaled syntax. For example, "10m" represents ten 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 using the **r** option in the configuration file.

Message timeouts

After sitting in the queue for a few days, a message will time out. This is to insure 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 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.

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 may 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.

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 defered (due to a host being down, etc.).

5 Normal message queueups.

6 Unusual but benign incidents, e.g., 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.

File Modes

There are a number of files that may have a number of modes. The modes depend on what functionality you want and the level of security you require.

To suid or not to suid?

Sendmail 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 userid is zero; if so, it resets the userid and groupid 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 my alias database be writable?

At Berkeley we have the alias database (/usr/lib/aliases*) mode 666. There are some dangers inherent in this approach: any user can add him–/her–self to any list, or can "steal" any other user's mail. However, we have found users to be basically trustworthy, and the cost of having a read–only database greater than the expense of finding and eradicating the rare nasty person.

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.

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.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time **sendmail** starts up, rather than easy for a human to read or write. On the "future project" list is a configuration–file compiler.

An overview of the configuration file is given first, followed by details of the semantics.

The 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 sharp symbol ('#') 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 may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are:

S*n*

Sets the current ruleset being collected to *n*. If you begin a ruleset more than once it deletes the old definition.

R *lhs rhs comments*

The fields must be separated by at least one tab character; there may 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 may be selected from the entire ASCII set, but user–defined macros should be selected from the set of upper case letters only. Lower case 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 using the escape sequence $*x*.

C and F -- define classes

Classes of words may 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 may be given names from the set of upper case letters. Lower case 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:

Path	The pathname of the mailer
Flags	Special flags for this mailer
Sender	A rewriting set for sender addresses
Recipient	A rewriting set for recipient addresses
Argv	An argument vector to pass to this mailer
Eol	The end-of-line string for this mailer
Maxsize	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 in this spec 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* may 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 using the **-f** flag. These typically are "root," "uucp," and "network," but on some users it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

T*user1 user2*...

There may be more than one of these lines.

P -- precedence definitions

Values for the "Precedence:" field may 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 zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

The Semantics

This section describes the semantics of the configuration file.

Special macros, 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 from line
n	The name of the daemon (for error messages)
o	The set of "operators" in addresses
q	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, on our system these definitions are:

```
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 UNIX (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 hostname of this site
x	The full name of the sender
y	The id of 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 UNIX (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 I send to "bollard@matisse" 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 (e.g., for the "Message–Id:" field). The **$i** macro is set to the queue id on this host; if put into the timestamp line it can be extremely useful for tracking messages. The **$y** macro is set to the id of the terminal of the sender (if known); some systems like to put this in the Unix "From" line. The **$v** macro is set to be the version number of **sendmail**; this is normally put in timestamps and has been proven extremely useful for debugging. The **$w** macro is set to the name of this host if it can be determined. The **$c** field is set to the "hop count," i.e., 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 timestamps in the message.

The **$r** and **$s** fields are set to the protocol used to communicate with **sendmail** and the sending hostname; these are not supported in the current version.

Conditionals can be specified using the syntax:

```
$?x text1 $| text2 $.
```

This interpolates *text1* if the macro $x is set, and *text2* otherwise. The "else" ($|) clause may 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 hostnames.

The left hand side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

$*	Match zero 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, 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:

$*n*	Substitute indefinite token *n* from LHS
$>*n*	"Call" ruleset *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 may be used anywhere.

The $> *n* syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule.

The $# syntax should *only* be used in ruleset zero. It causes evaluation of the ruleset 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 may be omitted. The *mailer* and *host* must be a single word, but the *user* may be multi–part.

A RHS may also be preceeded by a $@ or a $: to control evaluation. A $@ prefix causes the ruleset to return with the remainder of the RHS as the value. A $: prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The $@ and $: prefixes may preceed a $> spec; for example:

```
R$+    $:$>7$1
```

matches anything, passes that to ruleset seven, and continues; the $: is necessary to avoid an infinite loop.

Semantics of rewriting rule sets

There are five rewriting sets that have specific semantics.

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

local–part@host–domain–spec

If no "@" sign is specified, then the host–domain–spec *may* be appended from the sender address (if the C flag is set in the mailer definition corresponding to the *sending* mailer). Ruleset three is applied by **sendmail** before doing anything with any address.

Ruleset zero is applied after ruleset three 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.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

Mailer flags etc.

There are a number of flags that may be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. These are detailed in Addendum C. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers.

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 ruleset zero.

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 may come up may 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**. This involves setting the options, defining a few critical macros, etc. Since these are described in other places, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. **Sendmail** does this in three subphases. Rulesets one and two are applied to all sender and recipient addresses respectively. After this, you may specify per-mailer rulesets for both sender and recipient addresses; this allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

Philosophy

The particular philosophy you choose will depend heavily on the size and structure of your organization. I will present a few possible philosophies here.

One general point applies to all of these 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

Berkeley is an example of a large site, i.e., more than two or three hosts. We have decided that the only reasonable philosophy in our environment is to designate one host as the guru for our site. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they can get away with. In addition, any information they do have should be hints rather than solid information.

For example, a typical site on our local ether network is "monet." 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," our master host. Ucbvax may determine that the host name is illegal and reject the message, or may 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 *may* 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, our 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 transfered via the ethernet from monet to ucbvax, then via the ethernet from ucbvax to ucbarpa, and then is submitted to the Arpanet. Although this involves some extra hops, we feel this is 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) may 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 insure 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 may be determined from the syntax.

Relevant issues

The canonical form you use should almost certainly be as specified in the Arpanet protocols RFC819 and RFC822. Copies of these RFC's are included in */usr/doc/sendmail.dir* as *rfc819.lpr* and *rfc822.lpr.*

RFC822 describes the format of the mail message itself. **Sendmail** follows this RFC closely, to the extent that many of the standards described in this document can not be changed without changing the code. In particular, the following characters have special interpretations:

```
< > ( ) " \
```

Any attempt to use these characters for other than their RFC822 purpose in addresses is probably doomed to disaster.

RFC819 describes the specifics of the domain–based addressing. This is touched on in RFC822 as well. Essentially, each host is given a name which is a right–to– left dot qualified pseudo–path, 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.

Beware when reading RFC819 that there are a number of errors in it.

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 of. 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 ruleset three. This will be the hardest part of the job. Beware of doing too much to the address in this ruleset, 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 ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The Berkeley configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets 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 may use a comma separated list of rwsets for sequential application of rules to an input; ruleset three is always applied first. For example:

```
1,21,4 monet:bollard
```

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

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 userid before calling the mailer. **Sendmail** must be running setuid to root for this to work.

If this mailer is local (i.e., will perform final delivery rather than another network hop) the "l" flag should be given. Quote characters (backslashes and " marks) 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 (i.e., 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."

Other flags are described in Addendum C.

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 (ruleset four) 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 ruleset four.

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) may be used.

Finally, an argv template is given as the E field. It may 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; ruleset ten should be applied to sender addresses in the message and ruleset twenty 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 between the words "mail" and "-d." The second mailer is called "ether," it should 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 ruleset eleven and recipient addresses by ruleset twenty– one. There is a 100,000 byte limit on messages passed through this mailer.

Addendum A
Command Line Flags

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). *Cnt* is incremented during processing, and if it reaches MAXHOP (currently 30) **sendmail** throws away the message with an error.

-Fname 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 [Neigus73, Postel74, Postel77]), 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 Addendum B.

There are a number of options that may be specified as primitive flags (provided for compatibility with **delivermail**). These are the e, i, m, and v options. Also, the f option may be specified as the -s flag.

Addendum B
Configuration Options

The following options may 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.

dx 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 zero exit stat always

F*n* The temporary file mode, in octal. 644 and 600 are good choices.

f Save Unix-style "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 may be in old format, i.e., spaces 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 Be super-safe when running things, i.e., 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.

Addendum C
Mailer Flags

The following flags may be set in the mailer description.

f The mailer wants a **-f** *from* flag, but only if this is a network forward operation (i.e., 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 userid 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 (e.g, a user's mail.cf file).

n Do not insert a UNIX-style "From" line on the front of the message.

l This mailer is local (i.e., 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 lin'.

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 Upper case should be preserved in user names for this mailer.

h Upper case 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-style "From" lines with the ugly 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 insures 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; although this is required in RFC821, many hosts do not process return paths properly.

I This mailer will be speaking SMTP to another **sendmail** -- as such it can use special protocol features. This option is not required (i.e., 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 ruleset 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
```

Addendum D
Other Configuration

There are some configuration changes that can be made by recompiling **sendmail**. These are located in three places:

md/config.m4
These contain operating-system dependent descriptions. They are interpolated into the Makefiles in the *src* and *aux* directories. This includes information about what version of UNIX you are running, what libraries you have to include, etc.

src/conf.h
Configuration parameters that may be tweaked by the installer are included in conf.h.

src/conf.c
Some special routines and a few variables may be defined in conf.c. For the most part these are selected from the settings in conf.h.

Parameters in md/config.m4

The following compilation flags may be defined in the *m4CONFIG* macro in *md/config.m4* to define the environment in which you are operating.

V6
If set, this will compile a version 6 system, with 8-bit user id's, single character tty id's, etc.

VMUNIX
If set, you will be assumed to have a Berkeley 4BSD or 4.1BSD, including the **vfork**(2) system call, special types defined in <sys/types.h> (e.g, u_char), etc.

If none of these flags are set, a version 7 system is assumed.

You will also have to specify what libraries to link with **sendmail** in the *m4LIBS* macro. Most notably, you will have to include if you are running a 4.1BSD system.

Parameters in src/conf.h

Parameters and compilation options are defined in conf.h. Most of these need not normally be tweaked; common parameters are all in *sendmail.cf*. However, the sizes of certain primitive vectors, etc., are included in this file. The numbers following the parameters are their default value.

MAXLINE [256]
The maximum line length of any input line. If message lines exceed this length they will still be processed correctly; however, header lines, configuration file lines, alias lines, etc., must fit within this limit.

MAXNAME [128]
The maximum length of any name, such as a host or a user name.

MAXFIELD [2500]
The maximum total length of any header field, including continuation lines.

MAXPV [40]
The maximum number of parameters to any mailer. This limits the number of recipients that may be passed in one transaction.

MAXHOP [30]
When a message has been processed more than this number of times, **sendmail** rejects the message on the assumption that there has been an aliasing loop. This can be determined from the **-h** flag or by counting the number of trace fields (i.e, "Received:" lines) in the message header.

MAXATOM [100]
The maximum number of atoms (tokens) in a single address. For example, the address "eric@Berkeley" is three atoms.

MAXMAILERS [25]
The maximum number of mailers that may be defined in the configuration file.

MAXRWSETS [30]
The maximum number of rewriting sets that may be defined.

MAXPRIORITIES [25]
The maximum number of values for the "Precedence:" field that may be defined (using the **P** line in sendmail.cf).

MAXTRUST [30]
The maximum number of trusted users that may be defined (using the **T** line in sendmail.cf).

A number of other compilation options exist. These specify whether or not specific code should be compiled in.

DBM — If set, the "DBM" package in UNIX is used (see **dbm**(3X) in [UNIX80]). If not set, a much less efficient algorithm for processing aliases is used.

DEBUG — If set, debugging information is compiled in. To actually get the debugging output, the **-d** flag must be used.

LOG — If set, the *syslog* routine in use at some sites is used. This makes an informational log record for each message processed, and makes a higher priority log record for internal system errors.

QUEUE — This flag should be set to compile in the queueing code. If this is not set, mailers must accept the mail immediately or it will be returned to the sender.

SMTP — If set, the code to handle user and server SMTP will be compiled in. This is only necessary if your machine has some mailer that speaks SMTP.

DAEMON — If set, code to run a daemon is compiled in. This code is for 4.2BSD if the NVMUNIX flag is specified; otherwise, 4.1a BSD code is used. Beware however that there are bugs in the 4.1a code that make it impossible for **sendmail** to work correctly under heavy load.

UGLYUUCP — If you have a UUCP host adjacent to you which is not running a reasonable version of *rmail,* you will have to set this flag to include the "remote from sysname" info on the from line. Otherwise, UUCP gets confused about where the mail came from.

NOTUNIX — If you are using a non–UNIX mail format, you can set this flag to turn off special processing of UNIX–style "From " lines.

Configuration in src/conf.c

Not all header semantics are defined in the configuration file. Header lines that should only be included by certain mailers (as well as other more obscure semantics) must be specified in the *HdrInfo* table in *conf.c*. This table contains the header name (which should be in all lower case) and a set of header control flags (described below), The flags are:

H_ACHECK — Normally when the check is made to see if a header line is compatible with a mailer, **sendmail** will not delete an existing line. If this flag is set, **sendmail** will delete even existing header lines. That is, if this bit is set and the mailer does not have flag bits set that intersect with the required mailer flags in the header definition in *sendmail.cf,* the header line is *always* deleted.

H_EOH — If this header field is set, treat it like a blank line, i.e., it will signal the end of the header and the beginning of the message text.

H_FORCE — Add this header entry even if one existed in the message before. If a header entry does not have this bit set, **sendmail** will not add another header line if a header line of this name already existed. This would normally be used to stamp the message by everyone who handled it.

H_TRACE — If set, this is a timestamp (trace) field. If the number of trace fields in a message exceeds a preset amount the message is returned on the assumption that it has an aliasing loop.

H_RCPT — If set, this field contains recipient addresses. This is used by the **-t** flag to determine who to send to when it is collecting recipients from the message.

H_FROM — This flag indicates that this field specifies a sender. The order of these fields in the *HdrInfo* table specifies **sendmail'***s* preference for which field to return error messages to.

Let's look at a sample *HdrInfo* specification:

```
struct hdrinfo      HdrInfo[] =
{
            /* originator fields, most to least significant */

    "resent-sender",         H_FROM,
    "resent-from",           H_FROM,
    "sender",                H_FROM,
    "from",                  H_FROM,
    "full-name",             H_ACHECK,
        /* destination fields */
    "to",                    H_RCPT,
    "resent-to",             H_RCPT,
    "cc",                    H_RCPT,
        /* message identification and control */
    "message",          H_EOH,
    "text",             H_EOH,
        /* trace fields */
    "received",         H_TRACE|H_FORCE,

    NULL,               0,
};
```

This structure indicates that the "To:", "Resent- To:", and "Cc:" fields all specify recipient addresses. Any "Full-Name:" field will be deleted unless the required mailer flag (indicated in the configuration file) is specified. The "Message:" and "Text:" fields will terminate the header; these are specified in new protocols [NBS80] or used by random dissenters around the network world. The "Received:" field will always be added, and can be used to trace messages.

There are a number of important points here. First, header fields are not added automatically just because they are in the *HdrInfo* structure; they must be specified in the configuration file in order to be added to the message. Any header fields mentioned in the configuration file but not mentioned in the *HdrInfo* structure have default processing performed; that is, they are added unless they were in the message already. Second, the *HdrInfo* structure only specifies cliched processing; certain headers are processed specially by ad hoc code regardless of the status specified in *HdrInfo*. For example, the "Sender:" and "From:" fields are always scanned on ARPANET mail to determine the sender; this is used to perform the "return to sender" function. The "From:" and "Full-Name:" fields are used to determine the full name of the sender if possible; this is stored in the macro **$x** and used in a number of ways.

The file *conf.c* also contains the specification of ARPANET reply codes. There are four classifications these fall into:

```
char Arpa_Info[] =   "050"; /* arbitrary info */
char Arpa_TSyserr[] = "455"; /* some (transient) system error */
char Arpa_PSyserr[] = "554"; /* some (transient) system error */
char Arpa_Usrerr[] =  "554"; /* some (fatal) user error */
```

The class *Arpa_Info* is for any information that is not required by the protocol, such as forwarding information. *Arpa_TSyserr* and *Arpa_PSyserr* is printed by the *syserr* routine. TSyserr is printed out for transient errors, whereas PSyserr is printed for permanent errors; the distinction is made based on the value of *errno*. Finally, *Arpa_Usrerr* is the result of a user error and is generated by the *usrerr* routine; these are generated when the user has specified something wrong, and hence the error is permanent, i.e., it will not work simply by resubmitting the request.

If it is necessary to restrict mail through a relay, the *checkcompat* routine can be modified. This routine is called for every recipient address. It can return **TRUE** to indicate that the address is acceptable and mail processing will continue, or it can return **FALSE** to reject the recipient. If it returns false, it is up to *checkcompat* to print an error message (using *usrerr*) saying why the message is rejected. For example, *checkcompat* could read:

```
bool
checkcompat(to)
            register ADDRESS *to;
{
            if (MsgSize > 50000 && to->q_mailer != LocalMailer)
            {
```

```
                        usrerr("Message too large for non-local delivery");
                        NoReturn = TRUE;
                        return (FALSE);
                }
                return (TRUE);
        }
```

This would reject messages greater than 50000 bytes unless they were local. The *NoReturn* flag can be sent to supress the return of the actual body of the message in the error return. The actual use of this routine is highly dependent on the implementation, and use should be limited.

Addendum E
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.

References

[Birrell82] Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., "Grapevine: An Exercise in Distributed Computing." In Comm. A.C.M. 25, 4, April 82.

[Borden79] Borden, S., Gaines, R. S., and Shapiro, N. Z., The MH Message Handling System: Users' Manual. R-2367-PAF. Rand Corporation. October 1979.

[Crocker77a] Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., Standard for the Format of ARPA Network Text Messages. RFC 733, NIC 41952. In [Feinler78]. November 1977.

[Crocker77b] Crocker, D. H., Framework and Functions of the MS Personal Message System. R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.

[Crocker79] Crocker, D. H., Szurkowski, E. S., and Farber, D. J., An Internetwork Memo Distribution Facility -- MMDF. 6th Data Communication Symposium, Asilomar. November 1979.

[Crocker82] Crocker, D. H., Standard for the Format of Arpa Internet Text Messages. RFC 822. Network Information Center, SRI International, Menlo Park, California. August 1982.

[Metcalfe76] Metcalfe, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks", Communications of the ACM 19, 7. July 1976.

[Feinler78] Feinler, E., and Postel, J. (eds.), ARPANET Protocol Handbook. NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.

[NBS80] National Bureau of Standards, Specification of a Draft Message Format Standard. Report No. ICST/CBOS 80-2. October 1980.

[Neigus73] Neigus, N., File Transfer Protocol for the ARPA Network. RFC 542, NIC 17759. In [Feinler78]. August, 1973.

[Nowitz78a] Nowitz, D. A., and Lesk, M. E., A Dial-Up Network of UNIX Systems. Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. August, 1978.

[Nowitz78b] Nowitz, D. A., Uucp Implementation Description. Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. October, 1978.

[Postel74] Postel, J., and Neigus, N., Revised FTP Reply Codes. RFC 640, NIC 30843. In [Feinler78]. June, 1974.

[Postel77] Postel, J., Mail Protocol. NIC 29588. In [Feinler78]. November 1977.

[Postel79a] Postel, J., Internet Message Protocol. RFC 753, IEN 85. Network Information Center, SRI International, Menlo Park, California. March 1979.

[Postel79b] Postel, J. B., An Internetwork Message Structure. In Proceedings of the Sixth Data Communications Symposium, IEEE. New York. November 1979.

[Postel80] Postel, J. B., A Structured Format for Transmission of Multi-Media Documents. RFC 767. Network Information Center, SRI International, Menlo Park, California. August 1980.

[Postel82] Postel, J. B., Simple Mail Transfer Protocol. RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.

[Schmidt79] Schmidt, E., An Introduction to the Berkeley Network. University of California, Berkeley California. 1979.

[Shoens79] Shoens, K., Mail Reference Manual. University of California, Berkeley. In UNIX Programmer's Manual, Seventh Edition, Volume 2C. December 1979.

[Sluizer81] Sluizer, S., and Postel, J. B., Mail Transfer Protocol. RFC 780. Network Information Center, SRI International, Menlo Park, California. May 1981.

[Solomon81] Solomon, M., Landweber, L., and Neuhengen, D., “The Design of the CSNET Name Server.” CS–DN–2, University of Wisconsin, Madison. November 1981.

[Su82] Su, Zaw–Sing, and Postel, Jon, The Domain Naming Convention for Internet User Applications. RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.

[UNIX83] The UNIX Programmer's Manual, Seventh Edition, Virtual VAX–11 Version, Volume 1. Bell Laboratories, modified by the University of California, Berkeley, California. March, 1983.

Chapter 11

Administrative Commands

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 specific to the BSD4.2-based portions of the DOMAIN/IX SR9 software. The chapter (and the commands) are labeled with the number 8, both to distinguish the commands from those in the *DOMAIN/IX Command References*, and to maintain some continuity with the numbering scheme of the original *UNIX Programmers' Manuals*. However, the pages are numbered in the form 11-*n*, where *n* is the command's page number in this chapter (11).

You must be logged in as the super-user to use many of these utilities.

Each command returns two status bytes when it terminates. One of these bytes, which explains the cause for the command's termination, is supplied by the system. (The system returns zero for normal termination). When a command terminates normally, it also supplies a status byte, which is usually zero to indicate successful execution and not zero to report any problems encountered. See **wait**(2) and **exit**(2) for detailed information.

GLOSSARY

Process ID

Each active process in the system is uniquely identified by a positive integer called a process ID. The range of this ID is from zero to 30,000.

Parent Process ID

A new process is created by a currently active process; see **fork**(2). The parent process ID of a process is the process ID of its creator.

Process Group ID

Each active process is a member of a process group that is identified by a positive integer called the process group ID. This ID is the process ID of the group leader. This grouping permits the signaling of related processes; see **kill**(2).

Tty Group ID

Each active process can be a member of a terminal group that is identified by a positive integer called the tty group ID. This grouping is used to terminate a group of related process upon termination of one of the processes in the group; see **exit**(2) and **signal**(2).

Real User ID and Real Group ID

Each user allowed on the system is identified by a positive integer called a real user ID.

Each user is also a member of a group. The group is identified by a positive integer called the real group ID.

An active process has a real user ID and real group ID that are set to the real user ID and real group ID, respectively, of the user responsible for the creation of the process.

Effective User ID and Effective Group ID

An active process has an effective user ID and an effective group ID that are used to determine file access permissions (see below). The effective user ID and effective group ID are equal to the process's real user ID and real group ID respectively, unless the process or one of its ancestors evolved from a file that had the set-user-ID bit or set-group-ID bit set; see exec(2).

Super-user

A process is recognized as a *super-user* process and is granted special privileges if its effective user ID is zero.

Special Processes

The processes with a process ID of zero and a process ID of one are special processes and are referred to as *proc0* and *proc1*.

Proc0 is the scheduler. *Proc1* is the initialization process (**init**). *Proc1* is the ancestor of every other process in the system and is used to control the process structure.

File Name.

Names consisting of up to 32 characters may be used to name an ordinary file, special file or directory.

These characters may be selected from the set of all character values excluding zero (null) and the ASCII code for / (slash).

Note that it is generally unwise to use *, ?, [, or] as part of file names, because of the special meaning attached to these characters by the Shell. See **sh**(1).

Path Name and Path Prefix

A path name is a null-terminated character string starting with an optional slash (/), followed by zero or more directory names separated by slashes, optionally followed by a file name.

More precisely, a path name is a null-terminated character string constructed as follows:

```
        <path-name>::=<file-name>|<path-prefix><file-name>|/
        <path-prefix>::=<rtprefix>|/<rtprefix>
        <rtprefix>::=<dirname>/|<rtprefix><dirname>/
```

where <file-name> is a string of 1 to 32 characters other than the ASCII slash and null, and <dirname> is a string of 1 to 32 characters (other than the ASCII slash and null) that names a directory.

If a path name begins with a slash, the path search begins at the *root* directory. Otherwise, the search begins from the current working directory.

A slash by itself names the root directory.

Unless specifically stated otherwise, the null path name is treated as if it named a non-existent file.

Directory

Directory entries are called links. By convention, a directory contains at least two links, **.** and **..**, referred to as **dot** and **dot-dot** respectively. **Dot** refers to the directory itself and **dot-dot** refers to **dot**'s parent directory.

Root Directory and Current Working Directory

Each process has associated with it a concept of a root directory and a current working directory for the purpose of resolving path name searches. A process's root directory need not be the root directory of the root file system.

File Access Permissions

Read, write, and execute/search permissions on a file are granted to a process if one or more of the following is true:

> The process's effective user ID is super-user.
>
> The process's effective user ID matches the user ID of the owner of the file and the appropriate access bit of the ''owner'' portion (0700) of the file mode is set.
>
> The process's effective user ID does not match the user ID of the owner of the file, and the process's group ID matches the group of the file and the appropriate access bit of the ''group'' portion (070) of the file mode is set.
>
> The process's effective user ID does not match the user ID of the owner of the file, and the process's effective group ID does not match the group ID of the file, and the appropriate access bit of the ''other'' portion (07) of the file mode is set.

Otherwise, the corresponding permissions are denied.

NAME

addroot – add a root ID

USAGE

/etc/addroot

DESCRIPTION

Addroot adds a privileged 'root' person ID to the network registry. It need only be done for network registries established with software supplied prior to Apollo Software Release 9.0 (SR9.0). Normally, this process is performed as part of the SR9 installation procedure. **Addroot** must be run before you add a 'root' log-in account (with **/com/edacct**) and/or before you use **chown**(1) to specify super-user ownership of files.

NOTES

After executing **addroot**, you should run **crpasswd**(8) to update the UNIX accounting files in */etc*.

Addroot will succeed only if it is executed by a process that has the appropriate access to the network registry.

FILES

*/registry/**	network registry
*/etc/passwd**	password files
/etc/group	group file

RELATED INFORMATION

DOMAIN System Command Reference

NAME

arcv – convert archive files to new format

USAGE

arcv *file*

DESCRIPTION

The **arcv** command converts archive files created by the **ar**(1) command from PDP_11 format to a portable DOMAIN/IX format. The conversion operates directly on the file, exchanging the new format information for the old. The command returns the message ''not archive format'' if *file* is not in the old archive format.

The first line of each archive file is a 'magic string'. The magic string of old format archive files is the number 0177545; new archive files have the first line ''!<arch>''.

The DOMAIN/IX format is equivalent to VAX-11/780 archive format.

FILES

*/tmp/v**	temporary copy created during conversion

RELATED INFORMATION

ar(1)

NAME

catman – format the files for this manual

USAGE

/etc/**catman** [**–p**] [**–n**] [**–w**] [**–x**] *section-number(s)*

DESCRIPTION

The **catman** command creates the preformatted versions of the on-line manuals from the raw input files. The command examines each manual page in the *section-numbers* specified on the command line and any pages with missing or out-of-date preformatted versions are recreated. If any new pages are formatted, **catman** recreates the */usr/lib/whatis* database, which contains the name, section number, and a brief description for each entry in the manuals.

If you don't specify *section-number(s)*, **catman** checks all sections of the on-line manual for missing or out-of-date pages.

OPTIONS

–n prevents creation of */usr/lib/whatis*.

–p prints whatever updating would be done, without doing it.

–w causes the command to create the */usr/lib/whatis* database, without reformatting any manual pages.

–x causes **catman** to use the **manx** macros (DOMAIN/IX additions to **man**) in formatting the files. The following pipeline is executed, where *file* is the name of the manual page source file(s).

```
tbl file(s) | eqn | nroff -manx | col | /etc/swapul
```

EXAMPLE

```
catman 123
```

creates any formatted files that are out-of-date or that don't exist, in manual sections 1, 2, and 3 only.

FILES

/usr/man/man?/.**	raw (nroff input) manual pages
/usr/man/cat?/.**	preformatted manual pages
/usr/lib/makewhatis	commands to make *whatis* database

RELATED INFORMATION

man(1)
whatis(1)
manx(7)

NAME

chown – change the owner of files

USAGE

/etc/**chown** [**–f**] *owner file(s)*

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 log-in name that can be found in the password file.

In order to simplify accounting procedures, only the super-user can change the owner of files.

OPTIONS

–f	suppresses error messages.

FILES

/etc/passwd	password files

RELATED INFORMATION

chgrp(1)
chown(2)

NAME

cron – clock daemon

USAGE

/etc/cron

DESCRIPTION

The **cron** command executes commands at the dates and times specified in the file */usr/lib/crontab*. Since **cron** does not 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* file consists of lines with six fields each; fields are separated by spaces or tabs. The first five fields contain integer patterns that specify 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 (0-6, 0 is Monday)

Each field may contain a number or numbers. Two numbers separated by a minus sign indicate an inclusive range. A list of numbers separated by commas means that **cron** will execute when any of those values is true. 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. An asterisk means that **cron** will execute at all legal values for that field.

The sixth field is a string that is executed by the Shell at the time(s) and date(s) specified. A percent character (%) anywhere in the sixth field is translated to a newline character. The Shell executes only the first line (up to a % or end of line) of this command field. **Cron** interprets any characters after a % as standard input to the command.

Cron checks the file */usr/lib/crontab* once a minute.

EXAMPLE

A line of the form

```
0 2 * * * /etc/crpasswd
```

in the */usr/lib/crontab* file on the master system administrator's node will run **crpasswd** at 2:00 AM every morning.

FILES

/usr/lib/crontab	file of times, dates, and commands to be run

NAME

crpasswd – create password and group files

USAGE

/etc/**crpasswd**

DESCRIPTION

The DOMAIN/IX command **crpasswd** creates UNIX password and group files from the DOMAIN network registry's person, project, organization (PPO) file and the site directory's ACCOUNT (ACCT) file. All changes to */etc/passwd* and */etc/group* must be made using **crpasswd**. The sole exception to this rule is the "shell" field of */etc/passwd*, which cannot be derived from information in the registry and must be added by hand. To prevent discrepancies between */etc/passwd* and */etc/passwd.map*, run **crpasswd** after modifying the shell field of */etc/passwd*.

The **crpasswd** command also creates the file */etc/passwd.map*. The system calls **getuid** and **getgid** use this file to map DOMAIN subject identifiers to user and group IDs (and vice versa).

You must invoke **crpasswd** each the time you update the network registry. Otherwise, **getuid** and **getgid** will return a value of -1 for users not entered in */etc/passwd* and */etc/group*. The system administrator will usually have **cron** run **crpasswd** once a day.

NOTES TO DOMAIN/IX USERS

Do not delete the /etc/passwd and /etc/**group** files before you run **crpasswd**, as **crpasswd** can create new entries in these files without affecting the existing ones. Deleting these files will have unpleasant effects.

Cron uses **setuid**(3) to run as "root," so that you can have **cron** run **crpasswd** to modify password and group files owned by "root."

RELATED INFORMATION

getuid(2)
getgid(2)
DOMAIN System Command Reference

NAME

crpty – create psuedo tty device entries

USAGE

/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*.

RELATED INFORMATION

pty(4)

NAME

cvtumap – convert name trees from SR8 to SR9 name mapping

USAGE

/etc/cvtumap [-l] [-9] *pathname*

DESCRIPTION

Cvtumap converts an Apollo Software Release 8 (SR8) naming tree to an SR9 equivalent. This command is meaningful only to those users who are upgrading from SR8 software.

At SR8, a limited set of non-alphanumeric characters was available for use in filenames. These characters were 'mapped' into an internal representation for storage in the naming system. The mapping was reversed when the names were read from the naming system to be reported, via the **read**(2) kernel function, to the calling program. One example of this mapping is the character '-' (dash), which was translated for internal storage into ':5' (colon, five).

At SR9, DOMAIN/IX supports the full set of ASCII characters as valid components of a UNIX filename, with the exception of slash and null. When upgrading from an SR8 system to SR9, use **cvtumap** to map all SR8 trees that have non-alphanumeric characters in their names to the SR9 format.

EXAMPLE

/etc/cvtumap */sys*

will apply **cvtumap** to the directory *sys*, changing any file or directory names to the SR9 naming scheme.

NOTES

Certain name characters, such as tilde (˜) and accent grave (‘), had special significance at SR8. By default, these characters are mapped and stored as normal component characters in the SR9 naming scheme. The NAMECHARS environment variable may be used to retain the special significance of these characters. Set NAMECHARS to the set of characters for which you desire to retain the special (SR8/non-UNIX style) semantics of these characters.

Cvtumap no longer supports a return from the SR9 naming scheme to the SR8 one.

The UNIXNAMES environment variable is no longer supported.

OPTIONS

–9 converts from SR8 to SR9 mapping. (Included for compatibility only; you may run **cvtumap** without it.)

–l lists filenames and subdirectory names affected, as they are converted.

RELATED INFORMATION

DOMAIN System Command Reference

NAME

fix_cache - repair acl cache hash chains

USAGE

/etc/fix_cache

DESCRIPTION

The **fix_cache** command restores the node's acl_cache hash chain, while retaining the information already in the acl_cache. The acl_cache is used to improve performance when converting AEGIS file protections (acl's) into UNIX protection bits. The use of **fix_cache** is essential when, for example, the */etc/password* file is removed and then recreated, thus making the node's acl_cache invalid. (See the note under **crpasswd** for information on this case.)

RELATED INFORMATION

flush_cache(8)

NAME

flush_cache - clear the node's acl_cache

USAGE

/etc/**flush_cache**

DESCRIPTION

The **flush_cache** command clears the node's acl_cache. The acl_cache is used to improve performance when converting AEGIS file protections (acl's) into UNIX protection bits. You will have to use **flush_cache** rather than **fix_cache** if the acl_cache has been irreparably corrupted.

Use this command as a last resort. After executing a **flush_cache**, you must reboot the node.

RELATED INFORMATION

fix_cache(8)

NAME

halt – stop the processor

USAGE

```
/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.

OPTIONS

–n	Prevent the sync before stopping.
–q	Cause a quick halt, without attempting a graceful shutdown.
–y	Halt the system from a dialup.

RELATED INFORMATION

reboot(8)

NAME

htable – convert NIC standard format host tables

USAGE

/etc/htable [**-c** *connected-nets*] [**-l** *local-nets*] *file*

DESCRIPTION

Htable is used to convert host files from the format specified by Internet RFC 810 to the format used by the network library routines. Three files are created in the current directory as a result of running **htable**: *hosts*, *networks*, and *gateways*. The *hosts* file is used by the **gethostent**(3N) routines to map host names to addresses. The *networks* file is used by the **getnetent**(3N) routines to map network names to numbers. The *gateways* file is used by the routing daemon to identify ''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 added to the beginning of the appropriate **htable** output file, without interpretation. This feature allows sites to maintain local aliases and entries which are not normally present in the master database.

OPTIONS

The following options are allowed.

-c *connected-nets* The argument is a comma-separated list of networks to which this host is directly connected. In this list, networks may be indicated by name or by internet-standard dot notation, as shown here.

```
/etc/htable -c arpanet,128.32,local-ether
```

Htable only includes gateways that are directly connected to one of the specified networks or 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**. Entries for these networks will be taken from the *localhosts* file. This allows *localhosts* to override any entries in the input file.

Htable is used in conjunction with the **gettable**(8C) program, which retrieves the NIC database from a host.

NOTE TO DOMAIN/IX USERS

You cannot connect to a foreign network without the DOMAIN COM-ETHERNET product. The tables used in *etc* must agree with the tables used by COM-ETHERNET.

We recommend using one NIC host table. Use **gettable**(8C) to place it in the file */sys/tcp/hostmap/hosts.txt*. Then edit the file */sys/tcp/hostmap/local.txt* to reflect local conditions. Then run **makehost.sh** to create host and gateway maps for the COM-ETHERNET product. **Makehost.sh** executes **htable** to create the */etc/hosts*, */etc/gateways*, and */etc/networks* files. (We recommend not using the files *localhosts*, *localnetworks*, or *localgateways*, since all the necessary information is included in */sys/tcp/hostmap/local.txt*.)

The version of DOMAIN TCP/IP available at SR9.5 includes a slightly different version of **htable** that supports the new subnet facility. There are no changes in the way the command appears to the user.

RELATED INFORMATION

gettable(8C)
Managing TCP/IP-Based Communications Products

NAME

lpc – line printer control program

USAGE

/etc/lpc [*command* [*argument*]]

DESCRIPTION

Lpc controls the operation of the line printer system. For each line printer configured in */etc/printcap*, **lpc** can:

- disable or enable the printer,
- disable or enable the printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status, and associated spooling queues and printer daemons.

With no arguments, **lpc** prompts for commands from the standard input. If the *command* argument is supplied, **lpc** interprets it, as well as any *arguments* to it. You can redirect standard input, and make **lpc** read commands from a file. You must be logged in as the super-user to run **lpc**.

COMMANDS

Commands may be abbreviated; the following is the list of recognized commands.

? [*command*(*s*)]

help [*command*(*s*)]

Print a short description of each *command* specified in the argument list, or, if no *command* is specified, a list of the recognized *commands*.

abort { *all* | *printer*(*s*) }

Terminate an active spooling daemon on the local host immediately; then disable printing (preventing new daemons from being started by **lpr**) for either all *printers* or those specified in the argument list. This commmand only works on the DOMAIN node that is running the line printer server (**lpd**(8)).

clean { *all* | *printer*(*s*) }

Remove all files beginning with ''cf'', ''tf'', or ''df'' from the specified *printer* queue(s) on the local machine.

enable { *all* | *printer*(*s*) }

Enable spooling on the local queue for the specified *printers*. This allows **lpr** to put new jobs in the spool queue.

exit

quit Exit from **lpc**.

disable { *all* | *printer(s)* }
Turn the specified *printer* queues off. This prevents new jobs from being put into the queue by **lpr**.

restart { *all* | *printer(s)* }
Attempt to start a new *printer* daemon. This is useful when some abnormal condition causes the daemon to stop unexpectedly and leave jobs in the queue. If that happens, **lpq** will report that no daemon is present.

start { *all* | *printer(s)* }
Enable printing and start a spooling daemon for the listed *printers*.

status [*all*] [*printer(s)*]
Display the status of daemons and queues on the local machine.

stop { *all* | *printer(s)* }
Stop a spooling daemon after the current job completes, then disable printing.

topq *printer* [*jobnums*] [*username(s)*]
Move the jobs, specified by either *jobnums* (job numbers) or *usernames*, to the top of the printer queue in the order listed.

NOTE TO DOMAIN/IX USERS

If **lpc** and **lpd** are run on different nodes, the **abort** function may not work properly. If a process on the node running **lpc** has the same process ID as the print daemon on the node running **lpd**, **lpc** will attempt to abort the process on its own node, rather than terminate the **lpd** daemon on the other node.

DIAGNOSTICS

?Ambiguous command	abbreviation matches more than one command
?Invalid command	no match was found
?Privileged command	command can be executed by root only

FILES

/etc/printcap	printer description file
*/usr/spool/lpd/**	server spool directories
*/usr/spool/lpd/**	spool directories
/usr/spool/lpd//lock*	lock file for queue control

RELATED INFORMATION

lpq(1)
lpr(1)
lprm(1)
printcap(5)
lpd(8)

NAME

lpd – line printer daemon

USAGE

/usr/lib/lpd [–l] [–L *logfile* **] [** *port #* **] [–m]**

DESCRIPTION

Lpd is the line printer daemon (spool area handler) and is normally invoked at boot time from a node's *'node_data/etc.rc* file. With DOMAIN/IX, you have the option of running **lpd** on one or more nodes. See NOTES TO DOMAIN/IX USERS below for details.

The **lpd** daemon makes a single pass through the **printcap**(5) file to find out about the existing printers and prints any files left from a previous 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.

Access control is provided in two ways. First, all print requests must come from one of the machines listed in the file */etc/hosts.equiv*. Second, if the ''rs'' capability, which restricts remote use of a printer, is specified in the **printcap** entry for the printer requested, **lpr** requests will only be honored for users with accounts on the machine with that printer. In the case of DOMAIN/IX, this means that use of a printer on the DOMAIN ring would be restricted to other nodes on that DOMAIN ring.

The file *lock* in each spool directory prevents multiple daemons from becoming active simultaneously, and stores 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 either files to be printed or other, 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.

If a file cannot be opened, a message will be placed in the log file, normally the console. (Under DOMAIN/IX, errors are usually diverted to the file */usr/adm/lpd-errs*.) **Lpd** will try up to 20 times to reopen a file, 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 daemons 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 file name 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).

KEY CHARACTERS

J Job Name. String to be used for the job name on the burst page.

C Classification. 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. String to be used as the title for **pr**(1).

H Host Name. Name of the machine where **lpr** was invoked.

P Person. 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. Name of a file to print which is already formatted.

l Like ''f'' but passes control characters and does not make page breaks.

p Name of a file to print using **pr**(1) as a filter.

t Troff File. The file contains **troff**(1) output (phototypesetter commands).

d DVI File. The file contains output in DVI (Stanford) format.

g Graph File. The file contains data produced by plot.

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. Name of the font file to use instead of the default.

2 Troff Font I. Name of the font file to use instead of the default.

3 Troff Font B. Name of the font file to use instead of the default.

4 Troff Font S. 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 (in ascii).

U Unlink. Name of file to remove once printing is done.

N File name. The name of the file which is being printed, or a blank for the standard input (when **lpr** is invoked in a pipeline).

OPTIONS

–l Cause **lpd** to log valid requests received from the network. This can be useful for debugging purposes.

port# Change the Internet port number used to rendezvous with other processes.

–L *logfile* Write error conditions on *logfile*, rather than on the system console.

–m Send mail to a local user if requested by **lpr**. Normally, such mail is addressed to *user@host*, but this option causes the address to be simply *user*.

NOTE TO DOMAIN/IX USERS

The DOMAIN/IX implementation of **lpd** includes an optional file */usr/spool/lpd/servername* that controls which machine runs the line printer daemon **lpd**. If the file is not present, any machine on the network can run **lpd**. Note, however, that only one **lpd** server should be run per */usr/spool/lpd* directory. This limitation exists because the **flock**(2) protocol used by **lpd** is only valid on a single node, not across the ring.

To allow only one machine to run **lpd**, create this file and place in it the TCP/IP host name of the machine that is to run **lpd**. If anyone attempts to start **lpd** on a machine other than the one specified in */usr/spool/lpd/servername*, **lpd** will fail and return an error message indicating that the daemon may only be run on that machine.

For a node to run **lpd** successfully, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

NOTES

If the **–m** option is not specified, **lpd** sends mail to *user@host*. The **sendmail** configuration file must, therefore, know about DOMAIN TCP hosts.

FILES

/etc/printcap	printer description file
*/usr/spool/lpd/**	spool directories
/usr/spool/lpd/servername	name of machine that should run **lpd**
/etc/hosts.equiv	lists machine names allowed printer access

RELATED INFORMATION

lpq(1)
lpr(1)
lprm(1)
printcap(5)
lpc(8)
rc(8)
Managing TCP/IP-Based Communications Products

NAME

mkdisk – create disk device descriptor files

USAGE

/etc/mkdisk [–S | –W | –F] [–c *number*] **[–d** *number*] **[–l** *number*] **[–r]** *devname*

DESCRIPTION

The **mkdisk** command creates the block and character special device files that describe the disk devices attached to a node.

DOMAIN/IX nodes currently support three types of disk device. The conventional name for each type of disk is as follows:

Winchester disks	wn
Storage modules	sm
Floppy disks	fl

By convention, the names assigned to block disk device files on a DOMAIN/IX system include the disk type name as listed above, concatenated with the drive number and a character denoting the logical volume number. Thus, for example, logical volume 1 of drive 0 of a storage module disk would conventionally be named */dev/sm0a*. In this case, the prefix "dev" is the name of the directory where all "special files" normally live; the "sm" indicates that this is a storage module disk; the "0" is the drive number, and the "a" indicates that this is logical volume 1 (since logical volumes on a physical drive are numbered from 1). Similarly, */dev/fl1b* would name logical volume 2 of floppy disk drive 1.

The UNIX operating system makes a distinction between block and "raw" (character) devices. Each disk has a block device interface that makes the device byte-addressable. There are also "raw" devices available. Some UNIX systems place restrictions on the use of raw devices (for example, on some systems, raw devices must be read or written 512 bytes at a time). No such restrictions currently exist on DOMAIN/IX raw devices; in general, raw and block devices can be used interchangeably on DOMAIN/IX.

Typically, there will be both a raw device descriptor file and a block device descriptor file for each disk on a DOMAIN/IX system. The names of the raw device descriptor file and the block device descriptor file are related: if the block device is named */dev/xx0a*, the raw device will be named */dev/rxx0a*, with the "r" representing "raw".

The **mkdisk** program creates block and character disk device descriptor files. If run with no options, **mkdisk** tries to figure out the disk type and unit numbers from the disk descriptor file name you supply, using the naming conventions described above.

OPTIONS

You may use **mkdisk** to create disk descriptor files whose names do not follow the conventions described above, by supplying the needed information in options. If an option is explicitly supplied, it overrides the information **mkdisk** deduces from the filename.

–S	The disk is a storage module.
–W	The disk is a Winchester.
–F	The disk is a floppy disk.
–c *number*	Controller number.
–d *number*	Drive number.
–l *number*	Logical volume number.
–r	The disk is raw.

EXAMPLE

/etc/mkdisk /dev/wn1c

creates a block device descriptor file for logical volume 3 of drive 1 on the Winchester disk.

DIAGNOSTICS

The diagnostics are intended to be self-explanatory.

RELATED INFORMATION

mount (1)
umount (1)

NAME

mkptnr – create DOMAIN/IX node-specific files for diskless node

USAGE

/etc/**mkptnr** [**–p***partner_name*] [**–d***diskless_node_id*]

DESCRIPTION

Mkptnr creates the files required for the operation of DOMAIN/IX on a diskless node in the node's *//partner_name/sys/node_data.diskless_node_id* directory. These files are accessed by symbolic links from the */etc* directory.

It creates null *etc.mnttab*, *etc.mtab*, and *etc.utmp* files, if these files do not exist. It copies the template versions of *etc.rc* and *etc.inetd.conf* if these files do not already exist in the directory. It also copies the template versions of *thishost* and *networks* if these files do not already exist.

OPTIONS

–p *partner_name*

Copy the files to the node specified by partner_name. If you omit this option, **mkptnr** writes the files to the the current node, based on the value of the / node entry directory.

–d *diskless_node_id*

Use the entered node_id to construct the directory name. If you omit this option, **mkptnr** uses the value of the NODEID environment variable.

EXAMPLES

The following example creates the DOMAIN/IX files in the node_data directory for node 8f3d. It creates the files on the current (or current partner) node.

```
% mkptnr -d 8f3d
Files will be created and copied in //partner_name_node/sys/node_data.8f3d
Creating etc.mnttab, etc.mtab, etc.utmp (if not existant).
Copying etc.rc template.
Copying etc.inetd.conf template.
Copying thishost template.
Copying networks template.
%
```

FILES

/etc/mnttab a link to `node_data/etc.mnttab

/sys/node_data[.node_id]/etc.mnttab
System V mounted device table

/etc/mtab	a link to `node_data/etc.mtab
/sys/node_data[.node_id]/etc.mtab	BSD4.2 mounted device table
/etc/utmp	a link to `node_data/etc.utmp
/sys/node_data[.node_id]/etc.utmp	Node log-in record file
/etc/rc	a link to `node_data/etc.rc
/sys/node_data[.node_id]/etc.rc	**run_rc** shell script
/etc/templates/etc.rc	*/etc/rc* template file
/etc/inetd.conf	a link to `node_data/etc.inetd.conf
/sys/node_data[.node_id]/etc.inetd.conf	*inetd* daemon configuration file
/etc/templates/inetd.conf	*inetd.conf* template file
/sys/node_data[.node_id]/thishost	configuration file containing the node's TCP/IP host name
/sys/tcp/thishost_template	*thishost* template file
/sys/node_data[.node_id]/networks	TCP/IP configuration file with host's Internet network names and numbers
/sys/tcp/networks_template	*networks* template file

NAME

mount, umount – mount and dismount file system

USAGE

/etc/mount [*special name* [**–r**]]

/etc/mount –a

/etc/umount special

/etc/umount –a

DESCRIPTION

The **mount** command announces to the system that a removable file system is present on the device *special*. The *name* becomes the name of the newly-mounted root. *Name* must not already exist. The **umount** command announces to the system that the removable file system previously mounted on device *special* is to be removed.

Both **mount** and **umount** maintain a table of mounted devices in */etc/mtab*. On DOMAIN systems, this table is a link to *'node_data/etc.mtab*. If invoked without an argument, **mount** prints the table.

Before you can successfully mount a file system onto a disk, the disk's volume must be initialized. This is accomplished by executing **/com/invol** on the disk. If the disk's volume has already been initialized, you need not repeat the **/com/invol** execution. (Type **/com/help invol** at the Bourne or C Shell prompt to receive more information about this command.)

OPTIONS

–a Attempt to mount (unmount) all of the file systems described in */etc/fstab*. In this case, *special* and *name* are taken from */etc/fstab*. The *special* filename from */etc/fstab* is the block special name. (This option applies to both **mount** and **umount**.)

–r Mount the file system as read-only. (This option applies to **mount** only.)

FILES

/etc/mtab	mount table (link to *'node_data/etc.mtab*)
/etc/fstab	file system table (link to *'node_data/etc.fstab*)

CAUTIONS

Physically write-protected filesystems, as well as those on magnetic tape, must be mounted read-only. If they are not, errors will occur when access times are updated, whether or not any explicit write is attempted.

If you mount a filesystem that contains unreadable or otherwise invalid data, unpredictable things will happen.

RELATED INFORMATION

mount(2)
mtab(5)
fstab(5)
mkdisk(8)

NAME

pac – printer/plotter accounting information

USAGE

/etc/pac [–P*printer*] [–p*price*] [–s] [–r] [–c] [*username(s)*]

DESCRIPTION

Pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. If any *username(s)* are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

OPTIONS

–P*printer* Causes accounting to be performed for the named *printer*. Normally, accounting is performed for the default printer (site dependent), or the printer specified by the value of the environment variable **PRINTER**.

–p*price* Causes the value *price* to be used for the cost in dollars instead of the default value of 0.02.

–c Causes the output to be sorted by cost; usually the output is sorted alphabetically by name.

–r Reverses the sorting order.

–s Causes the accounting information to be summarized on the summary accounting file; this is necessary because, on a busy system, the accounting file can grow by several lines per day.

FILES

/usr/adm/?acct	raw accounting files
/usr/adm/?_sum	summary accounting files

NAME

rc – boot time shell script

USAGE

[cps] **/etc/run_rc**

DESCRIPTION

Run_rc performs a setuid to ''root'' and runs the shell script *'node_data/etc.rc.* This script initializes */tmp* and */usr/tmp* and optionally starts various daemons (e.g., **inetd**, **lpd**, and so on).

Edit the script to tailor it for the node on which it is installed.

This script usually runs at boot time. If you include the following line in your DM startup file:

```
cps /etc/run_rc
```

etc.rc will be run every time you reboot your node.

NOTE

The */etc/rc* file is a variant link, which resolves to *'node_data/etc.rc.*

NOTES TO DOMAIN/IX USERS

Beginning at SR9.5, the *'node_data/etc.rc* must be owned by root, and have the setuid bit set. The **run_rc** program can execute a user-created file, *'node_data/etc.rc.local,* which may contain commands that do not need to run as root.

FILES

/etc/run_rc	program that runs *'node_data/etc.rc*
/sys/node_data/etc.rc	boot time shell script that runs as root
/sys/node_data/etc.rc.local	boot time shell script for user commands
/etc/rc	link to *'node_data/etc.rc*
/etc/rc.local	link to *'node_data/etc.rc.local*

RELATED INFORMATION

cron(1)
uucp(1C)
inetd(8C)
rwhod(8C)
routed(8C)

NAME

reboot – reboot the processor

USAGE

/etc/reboot [–n] [–q]

DESCRIPTION

The **reboot** command writes out cached pages to the disks and then reboots the processor. Since DOMAIN processors do not currently support autoboot, this command behaves exactly like **halt(8)**.

OPTIONS

–n	Prevent the sync before rebooting.
–q	Reboot quickly, without first shutting down running processes.

RELATED INFORMATION

reboot(2)
halt(8)

NAME

renice – alter priority of running processes

USAGE

/etc/renice *priority* [[**–p**] *pid*] [[**–g**] *pgrp*] [[**–u**] *user*]

DESCRIPTION

Renice changes the scheduling *priority* of one or more running processes. **Renice** will operate on processes identified by process IDs, process group IDs, or usernames. Running **renice** on a process group causes all processes in the group to have their scheduling priority altered. **Renice** followed by a username causes all processes owned by that user to have their scheduling priority altered.

By default, the processes to be affected are specified by their process IDs.

Users can increase the *priority* of their own processes, i.e., make them run slower, by setting *priority* to a positive integer in the range 0 to PRIO_MIN (20). Setting the *priority* to a negative integer in the range zero to PRIO_MAX (-20) causes processes to run more quickly.

Useful priorities are:
19 (the processes will run only when nothing else on the system wants to)
0 (the ''base'' scheduling priority)
negative numbers (to make processes run faster)

OPTIONS

–g forces **renice** to interpret the next number as the process ID of a group of processes to be affected.

–u allows you to specify all the processes of a certain user or users.

–p resets the interpretation to process IDs (the default). This may be necessary when several options are strung together on one command line (see below).

NOTES

If you give the process a negative priority, the process cannot be interrupted. To interrupt the process, you must change the priority to a number greater than zero.

Unless you are the super-user, you can't increase the scheduling priorities of your own processes, even if you were the one who decreased the priorities in the first place.

EXAMPLE

```
/etc/renice +1 987 –u daemon root –p 32
```

decreases the priority of process IDs 987 and 32, as well as that of all processes owned by users daemon and root.

FILES

/etc/passwd	file that maps user names to user IDs

RELATED INFORMATION

getpriority(2)
setpriority(2)

NAME

sendmail – send mail over the internet

USAGE

/usr/lib/sendmail [**flags**] [*address* ...]

newaliases

mailq

DESCRIPTION

Sendmail sends a message to one or more log-in names, routing the message over whatever networks are necessary.

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 a ^D, or to a line with a single dot, and sends a copy of the text to all of the addresses listed. It determines the network 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''.

FLAGS

Flags are:

–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.

–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. 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. (Not currently supported by DOMAIN/IX.)

–C*file* Use alternate configuration file.

–d*X* Set debugging value to *X*.

–F*fullname* Set the full name of the sender.

–f*name* Sets the name of the ''from'' person (i.e., the sender of the mail). **–f** can only be used by the special users *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.

–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.

–r*name* An alternate (obsolete) form of the **–f** flag.

–t Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for people to send to. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed.

–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* specifies an alternate alias file.

c queues messages, rather than connecting immediately to mailers that are considered ''expensive''.

d*x* sets the delivery mode to *x*. Delivery modes are ''i'' for interactive (synchronous) delivery, ''b'' for background (asynchronous) delivery, and "q" for queue only – i.e., actual delivery is made the next time the queue is run.

D tries 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* causes **sendmail** to use this mode when creating temporary files. (See **chmod**(1)).

f saves UNIX-style From lines at the front of messages.

g*N* uses *N* as the default group ID when calling mailers.

H*file* calls the SMTP help file.

i ignores a dot on a line by itself as a message terminator.

L*n* sets the log level.

m sends to "me" (the sender) also, if I am in an alias expansion.

o uses 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* selects the directory in which to queue messages.

r*timeout* sets the timeout on reads. If none is set, **sendmail** will wait indefinitely for a mailer.

S*file* saves statistics in the named file.

s restarts the queue file, even under circumstances where it is not strictly necessary.

T*time* sets the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

t*stz,dtz* sets the name of the time zone. *Stz* is standard time zone; *dtz* is daylight time zone.

u*N* sets the default user ID for mailers to *N*.

If the first character of the username is a vertical bar, the rest of the username is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep **sendmail** from suppressing the blanks between arguments.

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.

NOTES

Sendmail converts blanks in addresses to dots. This is not consistent with ARPANET mail protocol RFC733 (NIC 41952), but it is consistent with the new protocol (RFC822).

NOTES TO DOMAIN/IX USERS

If you are running **sendmail** as a daemon, 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

/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

RELATED INFORMATION

mail(1)
rmail(1)
DARPA Internet Request For Comments RFC819, RFC821, RFC822;

NAME

sup – set UNIX-style protection

SYNTAX

/etc/sup *directory* ...

DESCRIPTION

Sup modifies each named *directory* so that new files and subdirectories created within that directory will receive UNIX-style protection. Normally, new files and directories that are created from DOMAIN/IX programs (using **creat**(2), **open**(2), **mknod**(2), or *mkdir*(2)) receive UNIX-style protection, but files and directories created by non-DOMAIN/IX programs (including the Display Manager) receive the initial (default) ACL of the directory in which they are created. The **sup** command arranges that all objects created in a directory to which **sup** has been applied will receive UNIX-style protection, i.e., owner, group and world rights based on the identity of the creating process.

EXAMPLE

To apply **sup** recursively to an entire *directory* hierarchy, type (in a Bourne shell):

find *directory* **-type d -exec sup { } \;**

NOTE

The **sup** command should not be run on any installed software, for example, the */com*, */bin*, or */usr* directories, since **sup** will nullify the initial default ACL and thereby deny the DOMAIN *sys_admin* account rights to those directories (and any directories created under them later). This means that a *sys_admin* account will be unable to update that software. In effect, you will probably not wish to use **sup** on directory levels higher than users' home directories.

If you inadvertantly run **sup** on a directory, you can restore the initial default ACL using the **id** option to **edacl**. For example, to restore the *sys_admin* account to *directory*'s initial default ACL, use the following command.

edacl -a *%.sys_admin.%.%* **-all -id** *directory*

RELATED INFORMATION

chmod(1)
creat(2)
open(2)
chown(8)

NAME

swapul – rearrange underlining

USAGE

/etc/swapul

DESCRIPTION

Swapul reads lines from standard input, rearranges underlining so that the underlines follow a character in the output stream (instead of being preceded by them), and writes the resulting text to standard output.

RELATED INFORMATION

DOMAIN System Command Reference

NAME

sync – update the super-block

USAGE

/etc/sync

DESCRIPTION

The **sync** command executes the *sync* system primitive. You can call **sync** 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). See **sync**(2) for details on the system primitive.

The **sync** operation is not actually necessary on DOMAIN hardware, because the system buffers are automatically written to disk at shutdown. Nevertheless, we provide it in the interest of compatibility.

RELATED INFORMATION

sync(2)
fsync(2)
halt(8)
reboot(8)
update(8)

NAME

syslog – log systems messages

USAGE

/etc/syslog [**–m***N*] [**–f***name*] [**–d**]

DESCRIPTION

Syslog reads a datagram socket and logs each line it reads into a set of files described by the configuration file */etc/syslog.conf*. Syslog configures when it starts up or receives a hangup signal.

Each message is one line. A message can contain a priority code, marked by a digit in angle braces at the beginning of the line. Priorities are defined in <*syslog.h*>, as follows:

LOG_ALERT this priority should essentially never be used. It applies only to messages that are so important that every user should be aware of them, e.g., a serious hardware failure.

LOG_SALERT messages of this priority should be issued only when immediate attention is needed by a qualified system person, e.g., when some valuable system resource dissappears. They get sent to a list of system people.

LOG_EMERG Emergency messages are not sent to users, but represent major conditions. An example might be hard disk failures. These could be logged in a separate file so that critical conditions could be easily scanned.

LOG_ERR these represent error conditions, such as soft disk failures, etc.

LOG_CRIT such messages contain critical information, but which can not be classed as errors, for example, 'su' attempts.

LOG_WARNING issued when an abnormal condition has been detected, but recovery can take place.

LOG_NOTICE something that falls in the class of ''important information''; this class is informational but important enough that you don't want to throw it away casually. Messages without any priority assigned to them are typically mapped into this priority.

LOG_INFO information level messages. These messages could be thrown away without problems, but should be included if you want to keep a close watch on your system.

LOG_DEBUG it may be useful to log certain debugging information. Normally this will be thrown away.

It is expected that the kernel will not log anything below LOG_ERR priority.

The configuration file is in two sections separated by a blank line. The first section defines files that syslog will log into. Each line contains a single digit which defines the lowest priority (highest numbered priority) that this file will receive, an optional asterisk which guarantees that something gets output at least every 20 minutes, and a pathname. The second part of the file contains a list of users that will be informed on SALERT level messages. For example, the configuration file:

```
8/usr/spool/adm/syslog
3/usr/adm/critical

smith
jones
robinson
```

logs all messages priority 8 or higher into the file /usr/spool/adm/syslog; and all messages of priority 3 or higher into /usr/adm/critical. The users ''smith'', ''jones'', and ''robinson'' will be informed on any subalert messages.

The flags are:

–m Set the mark interval to *N* (default 20 minutes).

–f Specify an alternate configuration file.

–d Debug mode. Do not attempt to ignore signals. Log errors to */dev/log.*

To bring syslog down, send it a terminate signal. It logs that it is going down and then waits approximately 30 seconds for any additional messages to come in.

There are some special messages that cause control functions. ''<*>N'' sets the default message priority to *N*. ''<$>'' causes syslog to reconfigure (equivalent to a hangup signal). This can be used in a shell file run automatically early in the morning to truncate the log.

Syslog creates the file */etc/syslog.pid* if possible containing a single line with its process id. This can be used to kill or reconfigure **syslog.**

NOTES

LOG_ALERT and LOG_SUBALERT messages should only be allowed to privileged programs.

As currently implemented, **syslog** cannot deal with kernel error messages.

In the rare case that a per-node */etc/syslog.conf* is needed, */etc/syslog.conf* can be made a link to *'node_data/syslog.conf.*

RELATED FILES

/etc/syslog.conf	the configuration file
/etc/syslog.pid	the process id

NAME

systype – display version stamp

SYNTAX

/etc/systype *file*

DESCRIPTION

Systype displays the UNIX version stamp of the specified object file. Three columns are displayed in the output. The first contains the systype; one of the following:

none
any
sys3
sys5
bsd4.1
bsd4.2

The second column contains the module name, as found in the object file. The third is the name of the object file, as specified on the command line.

RELATED INFORMATION

DOMAIN System Command Reference

NAME

update – update the super-block periodically

USAGE

/etc/update

DESCRIPTION

The **update** program executes the sync(2) primitive every 30 seconds.

The sync operation is not actually necessary on DOMAIN hardware, because the system buffers are automatically written to disk at shutdown. Nevertheless, we provide it in the interest of compatibility.

RELATED INFORMATION

sync(2)
sync(8)

NAME

update_slave – update auxiliary system administrator's nodes

USAGE

update_slave *//master_nodeid //slave_nodeid* . . .
update_slave *//master_nodeid[/systype] //slave_nodeid[/systype]* . . .

DESCRIPTION

Update_slave is a DOMAIN/IX shell script which updates the */etc* directories, as well as other system administrative information, on auxiliary system administrator's nodes. *Master_nodeid* is the node name of the single master system administrator's node on the network. *Slave_nodeid* is the node name of any slave, or auxiliary, system administrator's node. Multiple auxiliary nodes may be specified on the same command line.

Because of the distributed nature of the DOMAIN/IX system, you may maintain only one master set of such files as */etc/passwd*, */etc/passwd.map*, and */etc/group* per network, on the network's single master system administrator's node. Otherwise, the mapping between information in the DOMAIN registries and information in these files will not always be accurate.

However, since two DOMAIN networks can be connected to operate as one internet, it may be necessary to have copies of the */etc* files and directories on both networks, in case the connection fails. In that case, the **update_slave** script is used to ensure that information in */etc* on the slave administrator's node agrees with that in */etc* on the master administrator's node. You might also wish to have slave system administrator's nodes if you have a very large network.

The **update_slave** script is normally run from **cron** on the master node, in a line of the following form.

```
0 6 * * * //master_nodeid/etc/update_slave //master_nodeid //slave_nodeid
```

It can also be run from a shell prompt, if necessary.

The *bsd4.2* version of this script also updates the network files in the */etc* directory, *hosts*, *hosts.equiv*, *gateways*, *networks*, *protocols*, and *services*.

It is possible that a master administrative node will run one *systype* of DOMAIN/IX, and a slave, another. The second case of the command, under **USAGE**, is used in this case. In the example that follows, the master node(//administrator) is running *bsd4.2*, and is updating a slave node (//auxiliary) running *sys5*.

```
/etc/update_slave //administrator/bsd4.2 //auxiliary/sys5
```

NAME

ver – change the version of Shell commands

USAGE

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 **systype** value, which specifies the UNIX version of commands being executed by the Shell. The second form allows you to set the UNIX version, to **systype**. Valid **systypes** are "sys5" and "bsd4.2".

The third form of the command causes the UNIX version of **command** specified by **systype** to be executed, with no permanent change to **systype**.

FILES

/etc/ver	used for the third form of the command

RELATED INFORMATION

DOMAIN System Command Reference

NAME

ftpd – DARPA Internet File Transfer Protocol server

USAGE

/etc/ftpd [**–d**] [**–t***timeout*]

DESCRIPTION

Ftpd is the DARPA (Defense Advanced Research Projects Agency) Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification.

The **ftp** server currently supports the following **ftp** requests; upper- and lowercase operate identically.

Request	*Description*
ACCT	specify account (ignored)
ALLO	allocate storage
APPE	append to a file
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory ("ls -lg")
MODE	specify data transfer *mode*
NLST	give name list of files in directory ("ls")
NOOP	do nothing
PASS	specify password
PORT	specify data connection port
QUIT	terminate session
RETR	retrieve a file
RNFR	specify rename-from filename
RNTO	specify rename-to filename
STOR	store a file
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 765 are recognized, but not implemented.

Ftpd interprets filenames according to the conventions used by **csh**(1), which allows you to use the following metacharacters: *?[]{}˜.

Ftpd authenticates a user according to three rules.

1) The user's name must be in the password database, */etc/passwd*, and the account must not have a null password. If the password is null, a user must supply one before the account can perform any file operations.

2) The user's name must not appear in the file */etc/ftpusers*.

3) If the user's name is "anonymous" or "ftp", an anonymous **ftp** account must be present in the password file (i.e., user "ftp"). In this case, the user may log in by specifying any password. (By convention, this password is the original host's name).

In the last case, **ftpd** takes special measures to restrict access privileges. The "ftp" subtree should be constructed with care to ensure system security; 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 that are to be accessible via the anonymous account in this directory.

OPTIONS

–d turns on debugging for each socket created (SO_DEBUG). With debugging enabled, the system traces all sent and received on a socket.

–t sets the inactivity timeout period to *timeout*. Otherwise, the **ftp** server will timeout an inactive session after 60 seconds.

NOTES

Commands cannot be aborted.

The ''anonymous'' account may compromise the security and/or stability of the system.

The server must run as the super-user so that it can create sockets with privileged port numbers. It maintains the effective user ID of the user who is logged in, and reverts to the super-user only when binding addresses to sockets.

NOTES TO DOMAIN/IX USERS

Ftpd, like several other of the UNIX daemons, is normally invoked at boot time by running the **inetd**(8C) command from the */etc/rc* file. Enable the daemon by uncommenting the appropriate line(s) in the */etc/inetd.conf* file.

RELATED INFORMATION

ftp(1C)
inetd.conf(4)
inetd(8C)

NAME

gettable – get NIC format host tables from a host

USAGE

/etc/gettable *host*

DESCRIPTION

Gettable obtains the NIC standard host tables from a ''nicname'' server. (''Nicname'' is specified in the services specifications.) The program requests the tables from the specified *host*. The tables are placed in the file *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 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 the format used by the network library look-up routines.

NOTES

The program does not support requests for only part of the host tables.

NOTE TO DOMAIN/IX USERS

This command is only useful in conjunction with the DOMAIN COM-ETHERNET product.

RELATED INFORMATION

htable(8)
Managing TCP/IP-Based Communications Products

NAME

ifconfig – configure network interface parameters

USAGE

/etc/**ifconfig** *interface [address] [parameters]*

DESCRIPTION

Ifconfig assigns an address to a network interface and/or configures network interface parameters. It 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. The *interface* parameter is a string of the form *name unit.* The *address* is either one of the host names listed in */etc/hosts*, or a DARPA Internet address expressed in the Internet standard "dot" notation.

OPTIONS

If invoked with no options, **ifconfig** displays the current configuration for the specified *interface.* (Only the super-user may modify the configuration of a network interface.)

The following *parameters* may be set with **ifconfig**:

up Mark an interface "up."

down Mark an interface "down." When an interface is "down," the system will not attempt to transmit messages through that interface.

trailers (Not supported on DOMAIN/IX.) Enable 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.

–trailers Disable the use of a "trailer" link level encapsulation. This option is always enabled on DOMAIN/IX systems.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This option is always enabled on DOMAIN/IX systems.

–arp (Illegal on DOMAIN/IX.) Disable the use of the Address Resolution Protocol.

EXAMPLE

The following example demonstrates a typical use of **ifconfig** on DOMAIN/IX systems, returning the state of interface dr0.

```
% /etc/ifconfig dr0
```

```
dr0: 192.9.10.10 flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
```

DIAGNOSTICS

Ifconfig may return messages indicating the specified interface does not exist, the requested address is unknown, or that an unprivileged user tried to alter an interface's configuration.

RELATED INFORMATION

rc(8), intro(4N), netstat(1)

NAME

inetd – Internet superdaemon

USAGE

/etc/inetd

DESCRIPTION

The **inetd** is a server-manager that invokes internet services, such as **ftpd**(8C) or **rlogind**(8C), as necessary. Since it is a single process, **inetd** efficiently manages all types of internet connections.

The following servers and daemons can only be invoked by **inetd**. (This represents a change from the SR9.0 DOMAIN/IX convention).

/etc/ftpd	DARPA File Transfer Protocol daemon
/etc/rexecd	Remote execution server
/etc/rlogind	Remote log-in daemon
/etc/rshd	Remote Shell server
/etc/telnetd	DARPA TELNET protocol server

The **inetd** configuration file, */etc/inetd.conf* is, in nearly all installations, a link to *‘node_data/etc.inetd.conf*, a per-node file. **Inetd** reads this file at boot time and, in some cases, after it gets a hangup signal. Details about this file may be found under **inetd.conf**(4).

When a connection-oriented service such as **telnet**(1) is requested, **inetd** creates a process running the necessary server and passes it the connection as standard input (file descriptor 0). The server calls **getpeername**(2), which returns the source host and port.

When a datagram arrives for a datagram-oriented service, **inetd** creates a process running the appropriate datagram-oriented server, and then passes it the socket (on which the pending datagram is being held) as file descriptor 0.

If the server is "single-threaded," it simply takes over the socket. If the server is "multi-threaded," it connects directly to the peer, freeing up the socket for continued use by **inetd**. The distinction between single- and multi-threaded servers must be noted, using the "wait" and "nowait" keywords, respectively, in *‘node_data/etc.inetd.conf.*

If **inetd** encounters a serious error (e.g., unknown service, socket I/O error, incorrect server pathname), it attempts to log the error via syslog(8). We recommend that you run a syslog daemon on your node, especially if you are debugging a new server.

EXAMPLE

We recommend invoking **inetd** in background mode via the */etc/rc* Shell script. If you include the lines

```
if [ -f /etc/inetd ]; then
        /etc/inetd &
fi
```

in your node's *‘node_data/etc.rc* file, the */etc/run_rc* program will start **inetd** at boot time (after first checking to see if the file exists). Old versions of *‘node_data/etc.rc* typically include lines that start individual internet server daemons. If your node's copy of *‘node_data/etc.rc* includes such lines, remove them. We copy in a correct version of */etc/rc*, including these lines, on installation (unless you have a diskless node).

FILES

/sys/node_data/inetd.conf	**inetd** configuration file
/etc/services	internet services database

RELATED INFORMATION

inetd.conf(4)
services(5)
protocols(5)
rc(8)
syslog(8)
ftpd(8C)
rexecd(8C)
rlogind(8C)
rshd(8C)
telnetd(8C)
Managing TCP/IP-Based Communications Products

NAME

rexecd – remote execution server

USAGE

/etc/rexecd

DESCRIPTION

Rexecd is the server for the **rexec**(3X) routine. The server allows remote execution of commands; it authorizes execution by means of usernames and encrypted passwords.

Rexecd listens for service requests at the port indicated in the ''exec'' service specification. 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. This string is interpreted as an ASCII number in base 10.

2) If the number received in step 1 is not zero, it is interpreted as the port number of a secondary stream to be used for *stderr*. A second connection is then created to this port on the client's machine.

3) The server retrieves a null-terminated username (maximum 16 characters) from the first socket.

4) The server then retrieves a null-terminated, encrypted password (maximum 16 characters) from the first socket.

5) The server interprets the next string it retrieves from the first socket as a command (null-terminated). The upper bound on the size of the system's argument list limits the length of the command.

6) **Rexecd** then validates the user, as is normally done at log-in time. If successful, **rexecd** then changes to the user's home directory, and sets the appropriate user and group protections.

If any of these steps fails, the connection is aborted and a diagnostic message is returned.

7) The server returns a null byte on the connection associated with *stderr* and the command line is passed to the user's normal log-in Shell. The Shell inherits the network connections established by **rexecd**.

NOTES

Indicating ''Login incorrect'' as opposed to ''Password incorrect'' is a weakness that allows people to probe a system for users with no passwords.

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 (zero 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 list (as configured into the system).

Login incorrect.
No password file entry for the username exists.

Password incorrect.
An incorrect password was supplied.

No remote directory.
The **chdir** command to the home directory failed.

Try again.
A **fork** by the server failed.

/bin/sh: ...
The user's log-in Shell could not be started.

NOTES TO DOMAIN/IX USERS

Rexecd, like several other of the UNIX daemons, is normally invoked at boot time by running the **inetd**(8C) command from the */etc/rc* file. Enabling the daemon by uncommenting the appropriate line(s) in the */etc/inetd.conf* file.

For a node to accept incoming **rexec**, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

RELATED INFORMATION

rexec(3X)
inetd.conf(4)
inetd(8C)

NAME

rlogind – remote log-in server

USAGE

/etc/**rlogind** [**–d**]

DESCRIPTION

Rlogind is the server for the **rlogin**(1C) program. The server allows remote log-in to a system, with authentication based on privileged port numbers.

Rlogind listens for service requests at the port indicated in the "log-in" service specification. When a service request is received, the following protocol is initiated:

1) The server checks the client's source port. If the port number is not in the range 0-1023, the server aborts the connection.

2) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(5)), the server aborts the connection.

Once the source port and address have been checked, **rlogind** allocates a pseudo terminal, and manipulates file descriptors so that the slave half of the pseudo terminal becomes the *stdin*, *stdout*, and *stderr* for a log-in process. The log-in process is an instance of the **login**(1) program, invoked with the **–r** option. The log-in process then authenticates the user as described in **rshd**(8C), but if automatic authentication fails, **rlogind** presents the standard log-in message, as if the user were attempting to log in to a standard terminal line.

The parent of the log-in process manipulates the master side of the pseudo terminal and operates as an intermediary between the log-in process and the client of the **rlogin** program. In normal operation, the packet protocol provides ^S/^Q type facilities and propagates interrupt signals to the remote programs. The log-in process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM".

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium.

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.

Hostname for your address unknown.
No entry in the host name database exists for the client's machine.

Try again.
A **fork** by the server failed.

/bin/sh: ...
The user's log-in Shell could not be started.

NOTES TO DOMAIN/IX USERS

Rlogind, like several other of the UNIX daemons, is normally invoked at boot time by running the **inetd**(8C) command from the */etc/rc* file. Enable the daemon by uncommenting the appropriate line(s) in the */etc/inetd.conf* file.

For a node to accept incoming **rlogin** commands, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

RELATED INFORMATION

rlogin(1C)
inetd.conf(4)
inetd(8C)

NAME

route – manually manipulate the routing tables

USAGE

/etc/**route** [–f][*command args*]

DESCRIPTION

The **route** program allows you to manipulate the Internet routing tables by hand. It normally is not needed, as the system routing table management daemon, **routed**(8C), should tend to this task.

You may use the following three commands with **route**: *add*, to add a route; *delete*, to delete a route; and *change*, to modify an existing route.

All commands have the following syntax:

```
/etc/route
command destination gateway [metric]
```

where *destination* is a host or network to which the route is ''to'', *gateway* is the gateway to which packets should be addressed, and *metric* is an optional count indicating the number of hops to the *destination.* If no *metric* is specified, **route** assumes a value of 0.

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a ''local address part'' of INADDR_ANY, then the route is assumed to be a route to a network; otherwise, it is assumed to be a route to a host.

If the route is to a destination connected via a gateway, the *metric* should be greater than zero. Initially, **route** trys to look up all symbolic names specified for a *destination* or *gateway* in the host name database, **hosts**(5). If this lookup fails, **route** looks for the name in the network name database, **networks**(5).

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.

NOTES

No change operation is implemented. To change a route, add the new route, then delete the old one.

DIAGNOSTICS

add %s: gateway %s flags %x
Specified route is being added to the tables; values printed are from the routing table entry supplied in the **ioctl** (2) call.

delete %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.

not in table
Attempted a delete operation on an entry that wasn't present in the tables.

routing table overflow
Attempted an add operation, but the system was low on resources and was unable to allocate memory to create the new entry.

RELATED INFORMATION

intro(4N)
routed(8C)

NAME

routed – network routing daemon

USAGE

/etc/routed [**–s**] [**–q**] [**–t**] [*logfile*]

DESCRIPTION

Routed is started at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol to maintain up-to-date kernel routing table entries.

In normal operation, **routed** listens on number socket 520 (decimal) 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** to find those directly connected interfaces configured into the system and marked ''up'' (the software loopback interface is ignored). If multiple interfaces are present, 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*. See **route**(8C).

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 as efficient 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 and generates a *response* packet to all directly connected hosts and networks. **Routed** waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables; this allows possible unstable situations to settle.

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 internet.

Hosts acting as internetwork routers supply their routing tables every 30 seconds to all directly connected hosts and networks.

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 which may not be identified using the SIOGIFCONF *ioctl*. 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 (i.e., they should have a **routed** process running on the machine). Passive gateways are maintained in the routing tables forever and information about them is included in any routing information transmitted. Active gateways are treated like network interfaces. Routing information is distributed to the gateway. If no routing information is received for a period of time, the associated route is deleted.

The */etc/gateways* file is made up of a series of lines, each in the following format:

< *net* | *host* > *name1 gateway name2 metric value* < *passive* | *active* >

The *net* or *host* keyword indicates if the route is to a network or specific host.

Name1 is the name of the destination network or host. This may be a symbolic name located in */etc/networks* or */etc/hosts*, or an Internet address specified in ''dot'' notation; see **inet**(3N).

Name2 is the name or address of the gateway to which messages should be forwarded.

Value is a metric indicating the ''hop count'' to the destination host or network.

The keyword *passive* or *active* indicates if the gateway should be treated as *passive* or *active* (as described above).

OPTIONS

–s forces **routed** to supply routing information whether it is acting as an internetwork router or not.

–q is the opposite of the –s option.

–t prints a list of all packets sent or received, 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. *Logfile* is interpreted as the name of a file in which **routed**'s actions should be logged. This log will contain information about any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

NOTES

The kernel's routing tables may not correspond to those of **routed** for short periods of time while processes that are using existing routes exit; the only remedy for this is to place the routing process in the kernel.

Routed does not currently listen to intelligent interfaces, such as an IMP, or to error protocols, such as ICMP, to gather more information.

NOTES TO DOMAIN/IX USERS

Routed is normally started on a node at boot time, by executing the **run_rc** command. Uncomment the appropriate lines in the */etc/rc* file. You will run the **routed** daemon only on nodes connected directly to an Ethernet.

For a node to run **routed**, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

The version of DOMAIN TCP/IP available at SR9.5 includes a slightly different version of **routed** that supports the new subnet facility. There are no changes in the way the command appears to the user.

FILES

/etc/gateways	for distant gateways

RELATED INFORMATION

rc(8)
run_rc(8)
route(8C)
"Internet Transport Protocols", XSIS 028112, Xerox System Integration Standard.

NAME

rshd – remote Shell server

USAGE

/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 the authentication based on privileged port numbers.

Rshd listens for service requests at the port indicated in the ''cmd'' service specification. 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 resulting 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 *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. If the address is associated with a host for which no corresponding entry exists in the host name database, the server aborts the connection.

5) The server retrieves a null terminated username (maximum 16 characters) on the initial socket. This name is interpreted as a user account to be used on the **server**'s machine.

6) The server retrieves a null terminated username (maximum 16 characters) on the initial socket. This name is interpreted as a user account to be used on the **client**'s machine.

7) The server interprets the next string, up to a null character, as a command to be passed to the Shell. The upper bound on the size of the system's argument !ist limits the length of the command.

8) **Rshd** then validates the user according to the following steps. The remote username 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 zero), 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 searched 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 connection associated with the *stderr* and the command line is passed to the normal log-in Shell of the user. The Shell inherits the network connections established by **rshd**.

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium.

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 (zero is returned in step 9 above upon successful completion of all the steps prior to the command execution).

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).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the username 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.

/bin/sh: ...
The user's log-in Shell could not be started.

NOTES TO DOMAIN/IX USERS

Rshd, like several other of the UNIX daemons, is normally invoked at boot time by running the **inetd**(8C) command from the */etc/rc* file. Enable the daemon by uncommenting the appropriate line(s) in the */etc/inetd.conf* file.

For a node to accept incoming **rsh** commands, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

RELATED INFORMATION

rsh(1C)
rcmd(3X)
inetd.conf(4)
inetd(8C)

NAME

rwhod – system status server

USAGE

/etc/rwhod

DESCRIPTION

Rwhod is the server which 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.

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*.

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 {
                str'ct  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 a **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.

Status messages are generated approximately once every 60 seconds.

NOTES

Rwhod does not relay status information between networks. Consequently, people often interpret the server's dying as a machine in the network going down.

NOTES TO DOMAIN/IX USERS

Rwhod, like other DOMAIN/IX daemons, is invoked at boot time by the **run_rc** command. To run the **rwhod** on your node, uncomment the appropriate lines in the */etc/rc* file.

For a node to accept incoming **rwho** commands, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

Rwhod locks and unlocks the registry about once a minute. Running more than one **rwhod** process on a ring will cause a great deal of contention for the registry. Therefore, run **rwhod** only on the node connected directly to the Ethernet.

Rwhod reports statistics for the entire ring, not just the node it runs on.

RELATED INFORMATION

rwho(1C)
ruptime(1C)
rc(8)
run_rc(8)

NAME

talkd – server for **talk**(1) program

USAGE

/etc/talkd

DESCRIPTION

The **talkd** server, used by the **talk**(1) program, listens at the "udp" socket indicated in the "talk" service description. The conversation generated using **talk**, however, occurs over a TCP connection.

Refer to **services**(5) for more information about the "talk" service description.

We recommend invoking **talkd** in background mode via the */etc/rc* Shell script. If you include the lines

```
if [ -f /etc/talkd ]; then
        /etc/talkd &
fi
```

in your node's *'node_data/etc.rc* file, the */etc/run_rc* program will start **talkd** 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.

For a node to accept incoming **talk** commands, it must be correctly configured to run DOMAIN/IX TCP/IP. See *System Administration for DOMAIN/IX bsd4.2* for information on configuring TCP/IP.

RELATED INFORMATION

services(5)
talk(1)
inetd(8C)

NAME

telnetd – DARPA TELNET protocol server

USAGE

/etc/telnetd [–d] [*port*]

DESCRIPTION

Telnetd is a server for the DARPA standard TELNET virtual terminal protocol. The TELNET server operates at the port indicated in the ''telnet'' service description; see **services**(5). This port number may be overridden (for debugging purposes) by specifying a port number on the command line.

OPTIONS

–d enables debugging on each socket created by **telnetd** (see SO_DEBUG in **socket**(2)).

Telnetd operates by allocating a pseudo-terminal device for a client, then creating a log-in process that has the slave side of the pseudo-terminal as *stdin*, *stdout*, and *stderr*. **Telnetd** manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and log-in process.

When a TELNET session is started up, **telnetd** sends a TELNET option to the client side indicating a willingness to perform ''remote echo'' of characters. The pseudo terminal allocated to the client is configured to operate in ''cooked'' mode, and with XTABS and CRMOD enabled. Aside from this initial setup, the only mode changes **telnetd** will carry out are those required for echoing characters at the client side of the connection.

Telnetd supports binary mode, and most of the common TELNET options. It does not, however, support all of them. Consult the source code for an exact list of which options are implemented.

NOTES

See *Using telnet and ftp* for complete information on using the **telnet** command.

NOTES TO DOMAIN/IX USERS

Telnetd, like several other of the UNIX daemons, is normally invoked at boot time by running the **inetd**(8C) command from the */etc/rc* file. Enable the daemon by uncommenting the appropriate line(s) in the */etc/inetd.conf* file.

RELATED INFORMATION

telnet(1C)
inetd.conf(4)
inetd(8C)
Using telnet and ftp

NAME

tftpd – DARPA Trivial File Transfer Protocol server

USAGE

/etc/**tftpd** [**-d**] [*port*]

DESCRIPTION

Tftpd is a server which supports the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the "tftp" service description. This port number may be overridden (for debugging purposes) by specifying a port number on the command line.

OPTIONS

–d enables debugging on each socket created by **tftpd** (see SO_DEBUG in **socket**(2)).

The use of **tftp** does not require an account or password on the remote system. Because of this, **tftpd** will only allow users to access publicly readable files. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network; this definition may not be appropriate on all systems, and its implications should be considered before enabling **tftp** service.

NOTES

This server is only self-consistent (i.e., it operates with the user TFTP program, **tftp**(1C)). It does not support any other program.

The search permissions of directories leading to the files accessed are not checked.

NAME

uuclean – uucp spool directory clean-up

USAGE

/usr/lib/uucp/uuclean [options]

DESCRIPTION

Uuclean will scan the **uucp** spool directory for files with a specified prefix and delete all those which are older than the specified number of hours.

Cron(8) usually starts this program.

OPTIONS

The following options are available.

–d*dir* searches the directory named *dir* for files.

–p*pre* scans for files with *pre* as the file prefix. Up to 10 **–p** arguments may be specified. A **–p** without any *pre* following will cause all files older than *time* (see below) to be deleted.

–n*time* deletes files older than *time* hours, if the prefix test is satisfied. (Default *time* is 72 hours)

–m sends mail to the owner of the file when it is deleted.

NOTES

Uuclean does not automatically recognize **uucp** subdirectories. Use **uuclean** with the **-d** option to clean subdirectories.

FILES

/usr/lib/uucp	directory with commands used by **uuclean** internally
/usr/lib/uucp/spool	spool directory

RELATED INFORMATION

uucp(1C)
uux(1C)

NAME

uusnap – show snapshot of the UUCP system

USAGE

uusnap

DESCRIPTION

Uusnap displays a tabular synopsis of the current **uucp** activity on the system. The format of each line is:

```
site   N Cmds   N Data   N Xqts   Message
```

Where ''site'' is the name of the site with work, ''N'' is a count of each of the three possible types of work (command, data, or remote execute), and ''Message'' is the current status message for that site as found in the STST file.

''Message'' may include the time left before **uucp** can retry the call, and the number of times that **uucp** has tried to reach the site.

RELATED INFORMATION

uucp(1C)

NAME

writed – daemon for **write**(1) program

USAGE

/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* 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 */etc/run_rc* program 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*)

RELATED INFORMATION

write(1)
utmp(5)
rc(8)
run_rc(8)

Index

The letter *f* means "and the following page"; the letters *ff* mean "and the following pages". Symbols are listed at the beginning of the index. Entries in color indicates procedural information.

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: *System Administration for DOMAIN/IX BSD4.2*
Order No.: 009355 Revision: 00 Date of Publication: December, 1986

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: *System Administration for DOMAIN/IX BSD4.2*
Order No.: 009355 Revision: 00 Date of Publication: December, 1986

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