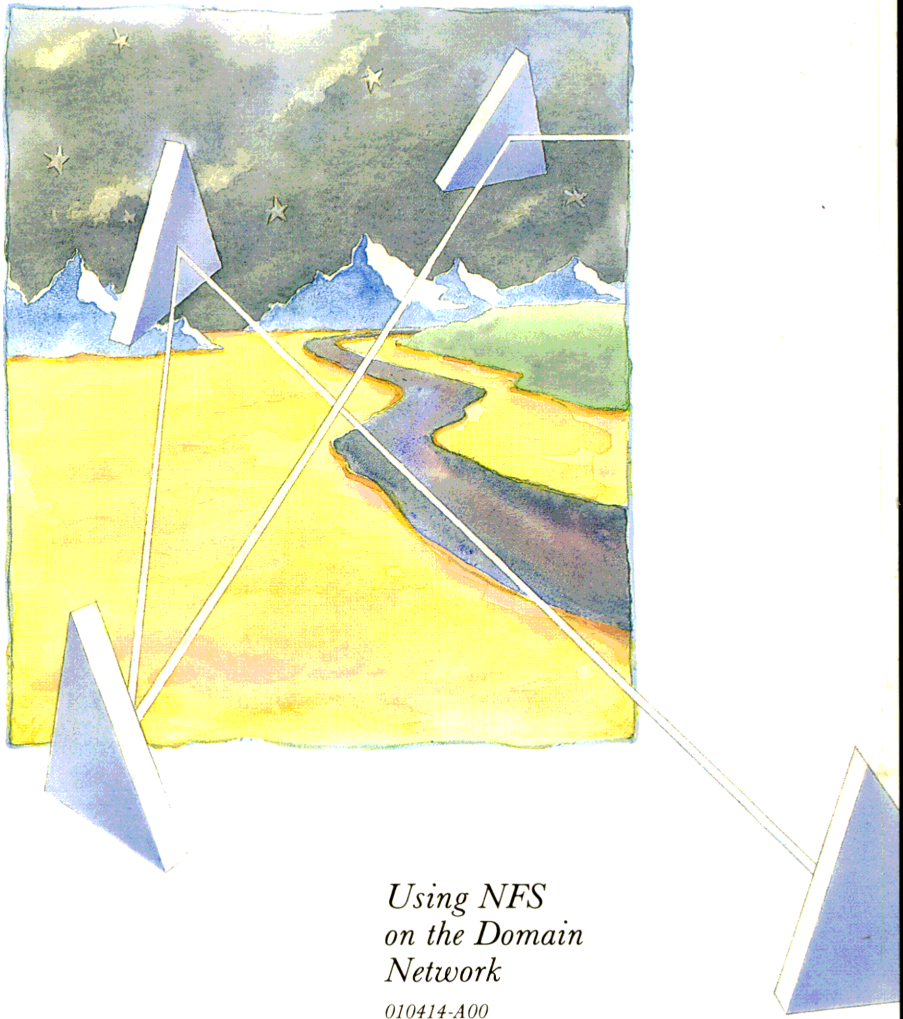

Using NFS on the Domain Network

010414-A00

apollo

Using NFS on the Domain Network

Order No. 010414-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: May, 1987
Latest Printing: June, 1988

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

Preface

Using NFS on the Domain Network describes the Domain® implementation of the Network File System (NFS*) protocol and explains how to use the software to access file systems, directories, and files located on foreign systems. The manual also tells users at foreign systems how to access the Domain/OS file system and its objects.

We assume that readers are already familiar with the operation and administration of Domain/OS and other UNIX* operating systems.

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

Organization of this Manual

We've organized this manual as follows:

Chapter 1	Provides an overview of the prerequisites, components, and operation of the Domain NFS software.
Chapter 2	Describes how to mount and unmount a remote file system and display NFS status information. The chapter also examines the differences between Domain and foreign pathnames and their impact on the NFS software.
Chapter 3	Identifies the Aegis, BSD, and SysV commands supported by the NFS software.
Chapter 4	Identifies the language I/O procedures and statements, functions, and system calls supported by the NFS software.
Appendix A	Describes the Domain NFS commands, server daemons, and reference files.
Appendix B	Tells the system administrator how to arrange NFS servers on the network and how to establish uniform user and group IDs.
Appendix C	Describes some common problems and their solutions.
Appendix D	Lists and explains the error messages for the **/etc/mount** and **/etc/umount** commands.

Related Manuals

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete list of related documents. For more information about topics discussed in *Using NFS on the Domain Network*, refer to the following documents:

- *Domain NFS Software Release Document* (010415)
- *Installing Software with Apollo's Release and Installation Tools* (008860)
- *Domain C Language Reference* (002093)
- *Domain FORTRAN Language Reference* (000530)
- *Domain Pascal Language Reference* (000792)
- *Domain/OS Call Reference, Volumes 1 and 2* (007196, 012888)
- *SysV Programmer's Reference* (005799)
- *BSD Programmer's Reference* (005801)
- *Programming with Domain/OS Calls* (005506)
- *Aegis Command Reference* (002547)
- *BSD Command Reference* (005800)
- *SysV Command Reference* (005798)
- *Using Your Aegis Environment* (0011021)
- *Using Your BSD Environment* (011023)
- *Using Your SysV Environment* (011022)
- *Managing Aegis System Software* (010852)
- *Managing BSD System Software* (010853)
- *Managing SysV System Software* (010851)
- *Using TCP/IP Network Applications* (008667)
- *Using the Open System Toolkit to Extend the Streams Facility* (008863)

Problems, Questions, and Suggestions

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

To get information about how to submit an APR, consult the appropriate *Command Reference* manual for your environment (Aegis, BSD, or SysV) and refer to the **mkapr** (make apollo product report) command description. You can view the same description online by typing

```
$ man mkapr (in the SysV environment)

% man mkapr (in the BSD environment)

$ help mkapr (in the Aegis environment)
```

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

Documentation Conventions

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

literal	Words or characters in bold type represent commands, keywords, or pathnames that you must use literally. In text, bold type can also indicate the first use of a new term.
symbolic	Words or characters in italic type are symbols for which you should subsitute a value.
`user input`	In examples, information that the user enters appears in color.
`system output`	In examples, output that the system displays appears in typewriter font.
[ ]	In command descriptions, square brackets enclose optional items.
{ }	In command descriptions, braces enclose a list from which you must choose an item.
|	In command descriptions, a vertical bar separates items in a list of choices.
——— ⌘ ———	This symbol indicates the end of a chapter.

Contents

Figures

Tables

Chapter 1

Overview

The Domain implementation of the NFS (Network File System) protocol allows users of Domain workstations and other systems running NFS software to transparently access each other's files. In this document, we use the term "Domain NFS" to distinguish the Domain NFS product from other implementations of the NFS protocol.

Domain NFS offers several advantages not found in other NFS implementations. The Domain NFS implementation lets users at **foreign systems** (that is, systems other than Domain systems) mount the Domain/OS network root directory (//) to gain access to files anywhere on a Domain network. Conversely, users at Domain workstations can mount foreign file systems in the Domain/OS network root directory, thereby enabling users throughout the Domain network to use the same names when accessing those file systems. Other NFS implementations do not recognize the concept of a network-wide root directory; they require a system to explicitly mount a **remote file system** before accessing its contents. (A remote file system is one that belongs to another system.)

Many NFS implementations also require you to remount remote file systems whenever you reboot a system. With Domain NFS, remote file systems remain mounted until you explicitly unmount them.

(However, Domain NFS does require you to unmount and then remount a remote file system if the TCP/IP address of its host changes.)

Many NFS implementations use a directory as a **mount point** for a remote file system. (A mount point is where a mounted file system joins another file system.) If the directory contains any objects at the time of the mount, the objects become unavailable and remain so until someone unmounts the remote file system and frees the directory. Domain NFS software uses a dedicated **gateway object** as a mount point (a gateway object is a special file that points to a remote file system), which means that your directories remain accessible.

1.1 Domain NFS Prerequisites

Domain NFS requires a particular network configuration and has several software prerequisites.

1.1.1 Configuration Requirements

The Domain workstations at your site must be connected to one or more foreign systems either through a gateway node or by being directly connected to the same network as the foreign systems.

In a **gateway node** configuration, Domain workstations are attached to an Apollo token ring network. The Apollo token ring network is in turn connected through a TCP/IP gateway node to another network (for example, an ETHERNET* network) to which foreign systems are attached. Figure 1-1 illustrates the gateway configuration.

* ETHERNET is a registered trademark of Xerox Corp.

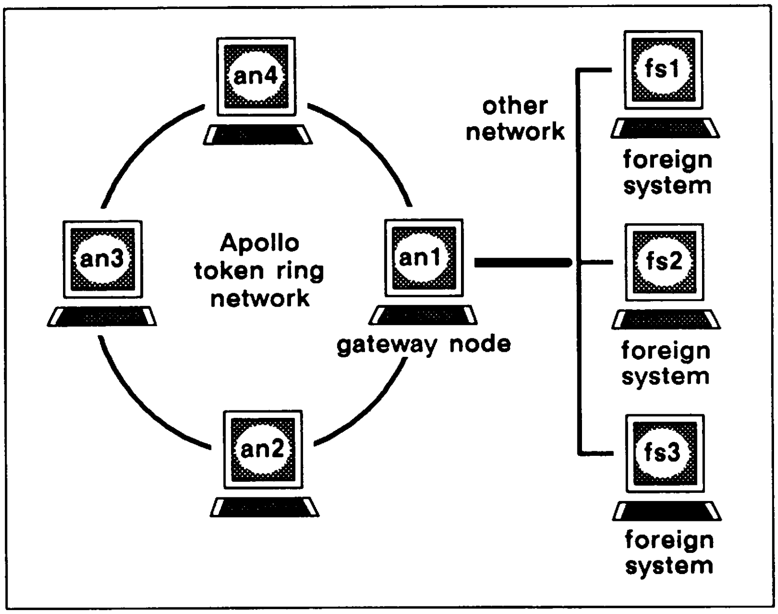

Figure 1–1. Gateway Configuration

In a **direct–connect** configuration, both Domain workstations and foreign systems are connected to the same network; again, an ETHERNET network is an example of a network to which both foreign systems and Domain workstations can be attached. Figure 1–2 illustrates the direct–connect configuration.

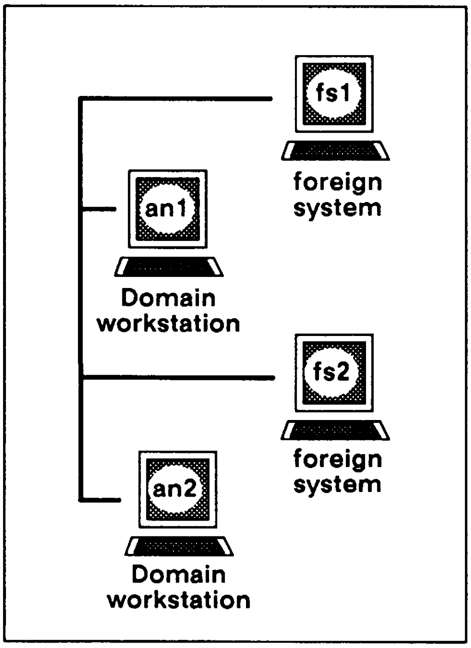

Figure 1-2. Direct-Connect Configuration

1.1.2 Software Requirements

You must have the following software installed on your workstation before you can install this version of NFS:

- Domain System Software Release 10 (SR10) or a later version
- The BSD operating environment, the SysV operating environment, or both (the BSD and SysV environments are part of standard Domain system software at SR10)
- TCP/IP software (TCP/IP is part of standard Domain system software at SR10)

See *Installing Software with Apollo's Release and Installation Tools* for instructions on installing this software. This book also describes how to install optional products. When you are ready to install NFS, you should first read the *Domain NFS Release Document*, then follow the installation procedure for optional products in *Installing Software*. System administrators should also read Appendix B of this manual for information about arranging servers on a network and about creating uniform user and group IDs.

1.2 The Domain NFS Components

In the remainder of this book, we refer to Domain workstations and foreign systems running NFS software as **hosts**. A **client host** uses resources provided by a **server host**.

Domain NFS software consists of server daemons, type managers, utility commands, and online **man** pages describing the commands.

The server daemons **mountd**, **nfsd**, and **portmap** (often referred to collectively as "the NFS server") run on a server host and thereby enable client hosts to mount its file systems. To be a server host, a Domain workstation must run all three server daemons.

The type managers **nfs_dir**, **nfs_file**, **nfs_dlink**, **nfs_flink**, and **nfs_gate** (referred to collectively as "the NFS type manager") reside in the **/sys/mgrs** directory of every Domain NFS host. A type manager intercepts requests for a specific type of object and determines under what circumstances that objects can be accessed. (For more information about type managers, see Section 2.4.3.) The type managers **nfs_dir**, **nfs_file**, **nfs_dlink**, and **nfs_flink** regulate access to remote directories, remote files, remote links to directories, and remote links to files, respectively; the type manager **nfs_gate** regulates access to Domain NFS gateway objects. When accessed from a Domain NFS host, gateway objects and objects on foreign hosts have the following type identifiers:

gateway object	**nfs_gate**
remote directory	**nfs_dir**
remote file	**nfs_file**
remote link to directory	**nfs_dlink**
remote link to file	**nfs_flink**

The following NFS utility commands reside in a host's /**etc** directory:

mount	mount a file system or directory
umount	unmount a file system or directory
showmount	display all remote mounts
nfsstat	display NFS statistics
rpcinfo	display NFS–related information

Appendix A in this manual describes these commands, plus the NFS server daemons and reference files.

To read the online **man** pages describing the commands, servers, and files, use the **man** command from within one of the UNIX shell process windows (BSD or SysV C shell, SysV Bourne shell, or BSD Korn shell).

1.3 How Domain NFS Software Works

All NFS implementations regard a host as being a client, a server, or both. A client host can mount a file system or a directory located on a server host; thereafter, the client host can access the mounted file system or directory as if it were a local file system or directory. In the Domain NFS implementation, a mounted file system or directory remains mounted until it is explicitly unmounted. (Hereafter, all references to file systems also include directories unless explicitly stated to be otherwise.)

Domain NFS permits you to mount and unmount a remote file system within BSD and SysV environments with the **/etc/mount** and **/etc/umount** commands, respectively. The command format used depends on the environment. The examples in this document present both BSD and SysV command format.

In the Domain NFS implementation, the **/etc/mount** command actually creates a gateway object that points to the file system. For example, the command line

```
$ /etc/mount -o soft sys1:/usr2/test /test          (BSD)

$ /etc/mount -f nfs,soft sys1:/usr2/test /test      (SysV)
```

creates the gateway object **/test** on the client host executing the command. The gateway object **/test** points to the file system **/usr2/test** on the server host **sys1** until you unmount the remote file system with the **/etc/umount** command.

To access an object within a mounted file system, simply include the name of the file system's gateway object in the pathname of the object. For example, assuming that you executed the **/etc/mount** command in the preceding paragraph, you can now use the pathname **/test/script** to access **/usr2/test/script** on host **sys1**. Once you have mounted a file system, the objects within it are accessible from any of the three Domain/OS environments (Aegis, BSD, or SysV); for example, the pathname **/test/script** will give you access to **/usr2/test/script** regardless of which environment you are using.

Similarly, you can set your working directory by using the gateway object as follows:

```
$ cd /test
```

A gateway object provides access to objects within its own file system; access to objects in other file systems requires additional gateway objects. For example, the slash (/) directory of a Domain NFS host constitutes a single file system; therefore, mounting the slash (/) directory of that host provides access to all of its objects.

However, as illustrated in Figure 1-3, mounting the slash (/) directory of other types of hosts may provide access to only some of the objects on those hosts. This is because users at other types of hosts sometimes divide the host's directory structure into several file systems, as shown in the figure. For example, to access all of the objects on the host in the figure, you must mount all of its file systems; mounting just the slash (/) file system does not provide access to the separate **/x** and **/y** file systems. This also means that you must use several gateway objects to access all of the host's objects.

Compare the mounting of the several file systems of a foreign host with the simplicity of mounting the Domain network. Users at foreign systems can mount an entire Domain network as if it were a single file system by mounting the Domain/OS network root directory. For example, a user at a foreign system could invoke the command:

```
$ /etc/mount chatham:// /apollo
```

to mount the Domain/OS network root directory as **/apollo**; in this example, the Domain host **chatham** serves as the intermediary between the foreign system and the Domain/OS network root directory. Should the user at the foreign system then want to access a Domain host such as the workstation **//harwich**, the user could use the pathname **/apollo/harwich**.

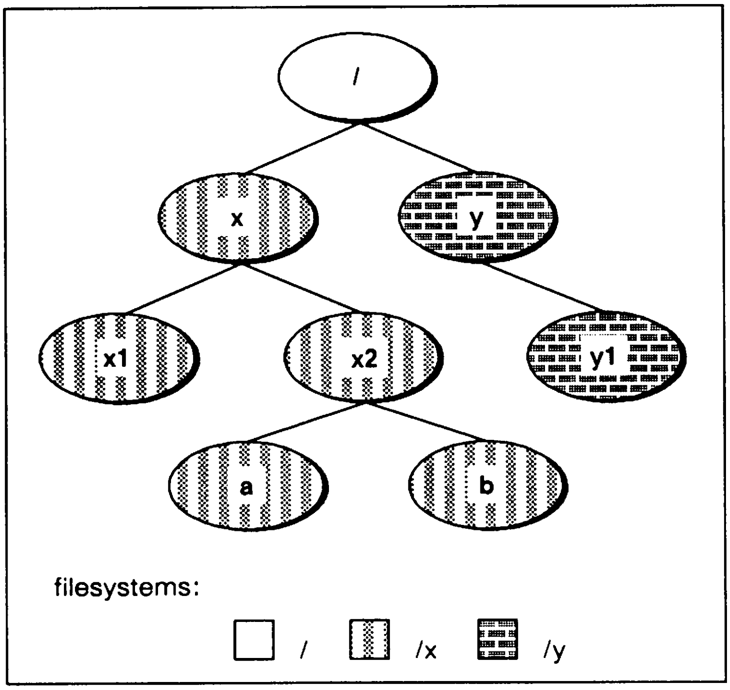

Figure 1-3. Foreign Host with Multiple File Systems

Chapter 2

Enabling Access to Remote File Systems

A Domain NFS host can mount and access file systems on any other NFS host. However, users of Domain workstations generally have no reason to use NFS software to access other Domain workstations since the network-wide Domain file system already provides more capabilities than those available via NFS software.

The following sections describe how to mount and unmount file systems from BSD and SysV environments, how to access files and directories, and how to request NFS statistics. Before executing the commands described in this chapter, you or your system administrator must first follow the procedures described in Appendix B and in the *Domain NFS Release Document*. For complete descriptions of the commands discussed in this chapter, see Appendix A.

2.1 Mounting a Remote File System

The **/etc/mount** command mounts a remote file system located on a server host. However, only the hosts authorized by the server's **/etc/exports** file can mount its file systems as clients. For more information about the **/etc/exports** file, see Appendix A or the **exports**(5) **man** page.

2.1.1 Using /etc/mount

When mounting a remote file system, use the **/etc/mount** command in the form

```
$ /etc/mount [nfsoptions] [host:fsname] [gwname]
```

in the BSD environment and

```
$ /etc/mount -f nfs[,nfsoptions] [host:fsname] [gwname]
```

in the SysV environment, where

nfsoptions	is one or more NFS mount options.
host:fsname	is the name of the server host, a colon (:), and the pathname of the file system to be mounted (for example, **host1:/usr**).
gwname	is the pathname of the gateway object (for example, **/host1_usr**).

The examples that follow show both BSD and SysV syntax for the **/etc/mount** command. If the syntax for a command line differs in the two environments, we identify the BSD syntax with the label "(BSD)" and the SysV syntax with "(SysV)".

If you execute the **/etc/mount** command without arguments, it displays every remote file system currently mounted by the host; the command retrieves this information from the list of mounted file systems it maintains in the file **/etc/mtab** (in the BSD environment) or **/etc/mnttab** (in the SysV environment). For example:

```
$ /etc/mount
lagoon:/beach/misc on /lagoon_misc type nfs (rw)
cataumet:/usr2/fish on /cataumet_fish type nfs (rw)
menemsha:/usr2/u/eel on /menemsha_eel type nfs (rw)
```

If you specify *host:fsname* and *gwname*, the command mounts the file system *host:fsname* and creates a gateway object named *gwname* that points to the file system. For example, the command

```
$ /etc/mount -o soft fleet:/usr2 /fleet_usr2          (BSD)

$ /etc/mount -f nfs,soft fleet:/usr2 /fleet_usr2      (SysV)
```

mounts the file system **/usr2** located on the server host **fleet** and creates the gateway object **/fleet_usr2**.

You must unmount and then remount a remote file system if the TCP/IP address of the host on which it resides changes.

For a complete description of **mount**, see Appendix A or the online **man** page for **mount**.

2.1.2 The soft and hard Options

We included the **soft** option in the last few sample **/etc/mount** commands for a particular reason. The **soft** and **hard** options control the way in which the **mount** command attempts to mount a file system and the way in which NFS attempts to access the objects on that file system once it has been mounted.

The **/etc/mount** command defaults to the **hard** option for compatibility with other NFS implementations. With the **hard** option, the command attempts to mount the target file system forever. If a network failure or some other condition makes the file system unavailable, the command ignores the error and simply continues to try to mount the file system. In addition, if you attempt to access an object on a file system that has been mounted with the **hard** option, NFS tries forever to gain access to that object, regardless of whether it is available. With the **soft** option, the command tries a number of times to mount the specified file system; if it remains unavailable after these attempts, the command reports an error and aborts. NFS operates similarly during attempts to access an object on a file system mounted with the **soft** option.

We recommend that you specify the **soft** option with **/etc/mount** when mounting a foreign file system from a Domain NFS host. However, if you are mounting a Domain file system from a foreign system, you should specify the **hard** option.

2.1.3 Mounting a Foreign File System in the Network Root Directory

Up to now, we've mounted remote file systems in the slash (/) directory of the client host; for example, **/fleet_usr2**. Actually, you can mount a remote file system in any directory to which you have access, including the network root directory (//).

Mounting a remote file system in the network root directory allows NFS hosts anywhere in the Domain network to use the same pathname when accessing the file system. For example, mounting the remote file system **fleet:/usr2** in the network root directory as **//fleet_usr2** enables all NFS hosts in the Domain network to access the file system by using the pathname **//fleet_usr2**.

With ns_helper Running on the Network

If the Domain network is running the naming server helper **ns_helper** (described in the *Managing System Software* books), use the **/etc/mount** command with its **root** option to mount a remote file system in the network root directory. Because **ns_helper** manages the global network root directory, it automatically makes the file system's name available to all Domain NFS hosts on the network. For example, the command

```
$ /etc/mount -o soft,root fleet:/usr //fleet_usr          (BSD)

$ /etc/mount -f nfs,soft,root fleet:/usr //fleet_usr     (SysV)
```

enables all Domain NFS hosts to gain access to the remote file system **fleet:/usr** via the pathname **//fleet_usr**. Note that you separate two or more options with commas and that the *gwname* argument must begin with two slashes (//).

When you use the **root** option, the **/etc/mount** command creates a gateway object in the local host's **'node_data** directory. The name of the gateway object consists of the command's *gwname* argument (without the two slashes) and the suffix **.nfs_root**, for example, **fleet_usr.nfs_root**.

Without ns_helper Running on the Network

If the Domain network is not running the naming server helper **ns_helper**, then mounting a remote file system in the network root directory requires more effort. In this case, the system administrator or the NFS users themselves must execute the same **/etc/mount** command on all Domain NFS hosts. This "manual" approach mounts the remote file system in the local copy of the network root directory located on every Domain NFS host. You should be running **ns_helper** if you have more than a few Domain workstations.

To mount a remote file system in the copy of the network root directory on your NFS host, execute the **/etc/mount** command with two slashes (//) before the *gwname* argument but without the **root** option. For example, the command

```
$ /etc/mount -o soft fleet:/usr //fleet_usr              (BSD)

$ /etc/mount -f nfs,soft fleet:/usr //fleet_usr          (SysV)
```

makes the remote file system **fleet:/usr** accessible to your NFS host as **//fleet_usr**. Executing this command on all Domain NFS hosts has the same effect as using the **root** option with the naming server **ns_helper** running.

When you mount a remote file system in a local copy of the network root directory, the **/etc/mount** command also creates a gateway object in the local host's **'node_data** directory. In this case, the name of the gateway object consists of the command's *gwname* argument (without the two slashes) and the suffix **.nfs**, for example, **fleet_usr.nfs**.

> **NOTE:** You can explicitly mount a remote file system in your local copy of the network root directory even when **ns_helper** is running. However, we strongly discourage this approach. Since the local copy serves as a cache of the global root directory when **ns_helper** is running, mounting a remote file system in the local copy can cause confusing and incorrect behavior.

As an alternative to executing the same **/etc/mount** command on all Domain NFS hosts, you can execute the command just once and have the Domain NFS users share the resulting mount point. For this approach to work, do not mount the remote file system in a copy of the network root directory; instead, create a mount point (that is, a gateway object) in a directory that everyone can access. For example, you could use the following command to mount the remote file system **fleet:/usr** and create the gateway object **fleet_usr** in the directory **//jib/gateways**:

```
$ /etc/mount -o soft fleet:/usr //jib/gateways/fleet_usr
                                                   (BSD)

$ /etc/mount -f nfs,soft fleet:/usr //jib/gateways/fleet_usr
                                                   (SysV)
```

Users at other Domain NFS hosts could then access the remote file system **fleet:/usr** via the pathname **//jib/gateways/fleet_usr**.

Naming Conventions

Whether you mount remote file systems in the network root directory with the **root** option or by the manual method, the result is the same: all Domain NFS hosts share the same names when accessing the file systems. System adminstrators therefore should develop naming conventions to prevent confusion and naming conflicts. We use the following naming convention when specifying the *gwname* argument of the **/etc/mount** command:

//host_filesystem

where *host* is the host name and *filesystem* is the file system name. For example, we use the *gwname* argument **//fleet_usr** when mounting the remote file system **fleet:/usr**. When mounting a remote root file system such as **fleet:/**, we simply use the *gwname* argument **//fleet**.

2.1.4 Mounting the Network Root Directory from a Foreign System

With most NFS implementations, users who want to access file systems on other NFS hosts must normally execute at least one **/etc/mount** command per host. However, an NFS user at a foreign system need only mount the Domain/OS network root directory (//) once to gain access to file systems located anywhere in the Domain network.

To mount the Domain/OS network root directory from a foreign system, an NFS user selects a Domain NFS server host and executes the **mount** command in the syntax that the foreign system supports, specifying the name of the Domain NFS server host and an existing directory on the foreign system. For example, suppose a user on a foreign BSD UNIX system selects the Domain NFS server host **landfall**, creates the directory **/apollo**, and executes the command

```
$ /etc/mount -o hard landfall:// /apollo
```

The command transforms the directory **/apollo** into a mount point for the Domain/OS network root directory. This means that the user at the foreign system can now access a file on any Domain workstation in the network, simply by substituting **/apollo/** for the double slashes (//) at the beginning of the file's pathname. For example, the user at the foreign system can now access the Domain file **//steamship/authority/summer_88/boat_sched** via the pathname **/apollo/steamship/authority/summer_88/boat_sched**.

NOTE: Avoid using the UNIX command string **ls -l /apollo** on a foreign system. This command causes the Domain NFS server to attempt to obtain status information about each host cataloged in the Domain/OS network root directory. If one or more of these hosts is unavailable (for example, because it has been powered down or is being serviced), this command will take a long time to complete. Also, the command will in some cases issue an error message stating that a host is not found when that host is, in fact, available. You can issue the UNIX command string **ls /apollo** from a foreign system without problem.

2.2 Unmounting a Remote File System

The **/etc/umount** command unmounts a remote file system mounted on a Domain/NFS host.

2.2.1 Using /etc/umount

The **/etc/umount** command has the same syntax for both BSD and SysV environments, which is

```
$ /etc/umount [ options ] { host:fsname | gwname }
```

where

options	is one or more command options.
host:fsname	is the name of the server host, a colon (:), and the pathname of the mounted file system (for example, **host1:/usr**).
gwname	is the pathname of the gateway object (for example, **/host1_usr**).

When you execute the command with the *gwname* argument, the command searches the contents of the file **/etc/mtab** (in the BSD environment) or **/etc/mnttab** (in the Sys V environment) and deletes the first entry it encounters that contains the *gwname* argument. For example, the following command unmounts the file system **/usr** on foreign host **fleet**.

```
$ /etc/umount /fleet_usr
```

The **/etc/umount** command also deletes the gateway object associated with the remote file system.

For a complete description of **umount**, see Appendix A or the online **man** page for **umount**.

2.2.2 Unmounting a Remote File System from the Network Root Directory

You can unmount a remote file system mounted in the Domain/OS network root directory in one of two ways, depending on whether the naming server **ns_helper** is running.

With ns_helper Running on the Network

If **ns_helper** is running, unmount the file system by using the **/etc/umount** command with its **-root** option. The **-root** option removes the file system's name from the network root directory and deletes the corresponding gateway object (for example, **'node_data/fleet_usr.nfs_root**). Note that you must execute this command on the Domain NFS host that originally mounted the remote file system; if you attempt to execute the command on another NFS host as in the following example, the command directs you to execute the command at the indicated NFS host:

```
$ /etc/umount -root //fleet_usr
umount: umount -root //fleet_usr must be made
 from node //falmouth
```

Without ns_helper Running on the Network

If **ns_helper** is not running, then use the **/etc/umount** command without the **-root** option; by default, the command unmounts the file system from the local copy of the network root directory. To unmount the remote file system from all copies of the network root directory, the system administrator or the NFS users themselves must execute the command on every Domain NFS host. For example, invoking the following command on every Domain NFS host:

```
$ /etc/umount //fleet_usr
```

unmounts the remote file system mounted as **//fleet_usr**. The command also deletes the gateway object **'node_data/fleet_usr.nfs** from all NFS hosts that execute the command.

2.3 Displaying Status Information

Besides displaying the remote file systems mounted on the local NFS host with the **/etc/mount** command, you can display NFS statistics with the **/etc/nfsstat** and **/etc/rpcinfo** commands. (The **nfsstat** and **rpcinfo** commands are generally used by system adminstrators when administering the network. For descriptions of these commands, see Appendix A.) You can also list the client hosts that have mounted file systems of a particular server host with the **/etc/showmount** command.

The **/etc/showmount** command lists the directories of a server host that are mounted by client hosts, displays all remote mounts in the form *host:fsname*, and lists all exported file systems (that is, file systems listed in the **/etc/exports** file of the server host).

The command takes the form

```
$ /etc/showmount [ options ] [ server_host ]
```

For explanation of all the available options, see Appendix A or the online **man** page for **showmount**.

By default, **showmount** lists the client hosts that have mounted file systems located on the local Domain NFS server host. You can also specify a different server host. For example, the command

```
$ /etc/showmount fleet
```

displays a list similar in form to the sample below:

```
$ /etc/showmount fleet
falmouth:
woods_hole:
nantucket:
```

2.4 Specifying Files on Remote File Systems

Wildcards and relative pathnames work regardless of whether you're specifying files located on a Domain workstation or on a foreign system running NFS software. However, a user at a foreign system cannot use certain extended pathnames or pathnames containing environment variables when accessing objects in Domain file systems. These features are unique to the Domain/OS operating system and are not recognized by foreign operating systems.

2.4.1 Wildcards and Relative Pathnames

You can use wildcards when specifying pathnames of objects in remote file systems. Simply follow the standard wildcarding procedures that apply to your shell.

For example, you can use the following shell command to list all objects starting with the characters **test** in the remote directory **//fleet/results**:

```
$ /com/ld //fleet/results/test?*            (Aegis)

$ /bin/ls //fleet/results/test*             (BSD or SysV)
```

Relative pathnames also apply to remote files and directories. For example, you can use the following shell commands to set your working directory to the remote directory **//fleet/results** and thereafter specify individual files by their leafnames:

```
$ wd //fleet/results                        (Aegis)
$ ld test?*

Directory "//harbor/sys/node_data/fleet.nfs_root":

test1   test2   test3   test4

4 entries listed.
$
```

```
$ cd //fleet/results                    (BSD or SysV)
$ ls test*
test1   test2   test3   test4
$
```

2.4.2 Special Symbols in Links

Links in the Domain/OS operating system can contain special symbols and environment variables in their text. For example, the following links include the '**node_data** and tilde (~) symbols and the UNIX **SYSTYPE** environment variable in their text:

```
memos         "~/memos"
/bin          "$(SYSTYPE)/bin"
/etc/exports "`node_data/etc/exports"
```

Foreign operating systems cannot translate pathnames containing such links because they do not know how to interpret these special symbols.

2.4.3 Typed Files

Most file systems support a small number of built-in file types. The Domain file system, on the other hand, supports an essentially unlimited number of file types; in fact, users can define new file types without changing the underlying file system. Associated with a given file type is a type manager that knows about the internal representation of files of that type. When a program attempts to access a file, the Domain file system automatically invokes the file's type manager to perform the requested operation. See *Using the Open System Toolkit to Extend the Streams Facility* for additional information about type managers.

The NFS protocol does not support the concept of typed files. All files are assumed to be unstructured streams of bytes. This does not generally cause problems for programs that are running on foreign systems and want to access Domain files, since their file requests go to the Domain NFS server, which uses the appropriate type manager to access the requested file. Thus, Domain files appear to be unstructured byte streams to a program running on a foreign system.

Many programs running on Domain systems also expect to create and manipulate unstructured files (that is, files of type **unstruct**). These programs can access the unstructured files on foreign systems through the NFS software without difficulty. In addition, there are a number of Domain file types that are equivalent to the **unstruct** type, for example, the **coff, bitmap** and **dp_dpd** file types. Because these **unstruct equivalent** files have no internal structure, they can be manipulated through Domain NFS as well. (For compatibility with earlier versions of Domain system software, Domain NFS treats the obsolete file types **usac** and **hdru** as unstruct equivalents.)

A Domain program that attempts through the NFS software to create a file that is not an unstruct equivalent on a foreign system will not be able to because the NFS protocol does not understand the concept of structured typed files. For example, the UNIX **cp** command and the Aegis **cpf** and **cpt** commands look at the type of the file being copied and attempt to create a target file of the same type. Since the NFS protocol only allows the commands to create unstructured files, they can copy only unstruct equivalent files to foreign systems.

When you copy an unstruct equivalent file to a foreign system, the file's type is lost. If you copy the file back to a Domain system, it is assigned the file type **unstruct**. You can restore the file to its original type with the Aegis **obty** command if you have kept a record of the old type. However, you *must not* restore original file types of **uasc** or **hdru**.

You can transfer a Domain file that is not an unstruct equivalent to a foreign system by redirecting the output of the Aegis **catf** command or the UNIX **cat** command, both of which read a file as an unstructured stream of bytes and write that byte stream to the target file. However, the resulting file on the foreign system has lost the internal structure of the original Domain file. It simply contains a stream of bytes representing the logical contents of the Domain file. For example, if **//dusk_usr** is a mount point for the **/usr** file system of a foreign system and **t.rec** is a file of type **rec** in your Domain working directory, then the UNIX command

```
$ cat t.rec > //dusk_usr/x.rec
```

creates a file named **x.rec** in the **/usr** file system of the foreign system and writes the logical contents of the Domain **t.rec** file to that file. The **t.rec** and **x.rec** files are not identical; the **x.rec** file does not have the internal structure of the Domain **t.rec** file.

You can also copy a Domain file that is not an unstruct equivalent to a foreign system by using that system's file copy command. However, the file's internal structure is lost as a result of this operation as well.

Note that the problems discussed in this section are not bugs but result from inherent limitations of the NFS protocol.

2.4.4 Extended Pathnames

In the Domain file system, a standard pathname consists of one or more directory names followed by an object name; for example, **//bud/tests/test1**. An extended pathname in the Domain file system is a standard pathname with an extension or **residual** at the end, for example, **//bud/tests/test1/12**.

Extended pathnames allow you to access all types of objects from within the Domain file system. For example, you can use extended pathnames to access the case history manager (casehm) files that the Domain Software Engineering Environment (DSEE™) creates. A casehm file includes the file's latest version and a history of all of its previous versions. To open and read a casehm file's most recent version, you use a standard pathname such as **//bud/tests/test1**. To open and read version 12 of the **test1** file, you would use the extended pathname **//bud/tests/test1/12**. For more information about extended pathnames, consult *Using the Open System Toolkit to Extend the Streams Facility*.

Most NFS implementations on foreign systems cannot process extended pathnames because most foreign systems expect every pathname component other than the rightmost to be a directory, and reject any pathname that does not follow this pattern.

Thus, in the above example of the casehm file, an NFS implementation on a foreign system that uses this kind of pathname processing will be unable to access any but the most recent version of **test1**. That is, you will be able to open and read **//bud/tests/test1** from the foreign system, but not **//bud/tests/test1/12**.

2.4.5 Super–User Limitation

The NFS protocol does not allow processes to have super–user privileges across networks. If a super–user process on a foreign system attempts to access a Domain file, it has the same access rights as a process with the subject identifier **user.none.none**. If a super–user process on a Domain workstation attempts to access a file on a foreign system, it has the same access rights as the least privileged user on that system. On many systems this corresponds to a user ID of –**2** and a group ID of –**2**.

Chapter 3

Accessing Files with Commands

With Domain NFS software, you can access files in mounted file systems through the Display Manager and through BSD, SysV, and Aegis shell commands.

3.1 Using the Display Manager

The Display Manager (DM) works the same way for files located on foreign hosts as it does for files located on Domain workstations.

Note that you cannot use the DM to create or manage processes on a foreign host. To perform such operations, you must log on to the foreign system with **telnet** or **rlogin** (consult *Using TCP/IP Network Applications* and either the BSD or SysV *Command Reference* for descriptions of these commands).

3.2 Using BSD and SysV Commands

The Domain NFS implementation allows you to use almost all of the standard BSD and SysV file access and manipulation commands on foreign files. The limitations are as follows:

- BSD or SysV commands cannot create or execute Domain object modules located on foreign systems.
- You cannot execute the following UNIX commands on files located on foreign systems:

  ld
  nm
  size
  strip

  The output of the **cc** compiler must be sent to a Domain workstation and not to a file located on a foreign system (although the source files can be on a foreign system).
- The **df** command is not supported.

You can use the UNIX shells' standard wildcards when accessing objects on a foreign system.

Consult the *BSD Command Reference* for descriptions of BSD commands and the *SysV Command Reference* for descriptions of SysV commands.

3.3 Using Aegis Shell Commands

The Domain NFS implementation lets you use most of the standard Aegis shell commands when working with remote files.

3.3.1 Special Characters in Command Lines

You can execute Aegis shell commands that include any of the special characters described in the *Aegis Command Reference*.

The special characters include

- Pathname wildcards (for example, ?, %, *, ..., =)
- Input/output control characters (for example, <, >>, |, ( ))
- Parsing operators (for example, #, ;, &, ^, !, @)

3.3.2 Aegis Shell Command Support

Except for the commands listed in the following tables, the Domain NFS implementation completely supports all Aegis shell commands that access user files.

Table 3–1 contains commands with restricted support, namely, the Domain language compilers. The compilers cannot save their object files on foreign systems.

Table 3–2 contains commands that currently do not work with Domain NFS.

Table 3–3 contains commands that are specific to the Domain/OS operating system. For example, the table contains the **crsubs** command which creates a protected subsystem.

In addition, Aegis commands cannot create or execute Domain object modules located on foreign systems.

Table 3-1. Restricted Aegis Shell Commands

Command	Description
cc	Domain C compiler
ftn	Domain FORTRAN compiler
pas	Domain Pascal compiler

Table 3-2. Currently Unsupported Aegis Shell Commands

Command	Description
bind	Combine object modules into an executable file
cpboot	Copy the system boot file sysboot
crf	Create a file
crpad	Create a transcript pad and window
debug	Invoke the Language Level Debugger
edfont	Edit a character font
edmtdesc	Edit magtape descriptor file
hpc	Program counter histogram
lbr	Combine object modules into a library
lvolfs	List free space on logical volumes
rbak	Restore/index magnetic media backup file
rwmt	Read/write foreign magtapes
wbak	Create a magnetic media backup file

Table 3-3. Domain/OS-Specific Aegis Shell Commands

Command	**Description**
acl	List or copy an Access Control List
chuvol	Change a volume's UID
cmt	Compare source tree to target tree
crsubs	Create a protected subsystem
ctob	Catalog an object
dmtvol	Unmount a logical volume (use umount)
edacl	Edit/list an Access Control List
ensubs	Enter a protected subsystem
find_orphans	Locate and catalog uncataloged objects
invol	Initialize disk volumes
lkob	Lock an object
llkob	List locked objects
mtvol	Mount a logical volume (use mount)
obty	Set or display the type of an object
salacl	Salvage an Access Control List
sald	Salvage a directory
salvol	Verify/correct allocation of disk blocks
subs	Set or display subsystem attributes
uctob	Uncatalog pathname but do not delete
ulkob	Unlock an object
xsubs	Run shell script subsystem manager

Chapter 4

Accessing Files from Programs

With the Domain NFS implementation, you can use the following programming techniques to access files located in foreign file systems:

- BSD or SysV system calls.
- The Aegis stream and name management system calls. These include **ios_$**, **stream_$**, and certain **name_$** calls.
- The standard file access and I/O procedures that are part of your programming language, for example, the Pascal **readln** or **put** procedures. Domain NFS software supports the standard I/O procedures of Domain C, FORTRAN, and Pascal.

  NOTE: This chapter lists calls and procedures that access and manipulate files and directories. It includes calls that take file or directory names as input or output arguments. It does not include calls that can only manipulate Domain system files or objects.

4.1 BSD and SysV System Calls

The Domain NFS implementation supports both BSD and SysV system calls. Table 4–1 lists these calls and indicates with check-marks (✓) whether each call is included in BSD, SysV, or both. The table also indicates whether each call is fully supported (Y), unsupported (N), or supported with restrictions (R). See the BSD or SysV *Programmer's Reference* for details about these calls and their use.

Table 4–1. BSD and SysV Call Support

Function	System		Support (Y/N/R)
	BSD	System V	
access	✓	✓	Y
chdir	✓	✓	Y
chmod	✓	✓	Y
chown	✓	✓	N
clearerr	✓	✓	Y
close	✓	✓	Y
closedir	✓		Y
creat	✓	✓	Y
ctermid		✓	Y
default_acl	✓	✓	N
dup	✓	✓	Y
dup2	✓		Y

(continued)

Table 4-1. BSD and SysV Call Support (continued)

Function	System BSD	 System V	Support (Y/N/R)
execl	✓	✓	N
execle	✓	✓	N
execlp	✓	✓	N
execv	✓	✓	N
execve		✓	N
execvp	✓	✓	N
fclose	✓	✓	Y
fcntl	✓	✓	Y
fdopen	✓	✓	Y
feof	✓	✓	Y
ferror	✓	✓	Y
fflush	✓	✓	Y
fgetc	✓	✓	Y
fgets	✓	✓	Y
fileno	✓	✓	Y
flock	✓		N
fopen	✓	✓	Y
fprintf	✓	✓	Y
fputc	✓	✓	Y
fputs	✓	✓	Y
fread	✓	✓	Y

(continued)

Table 4-1. BSD and SysV Call Support *(continued)*

Function	**System** BSD	System V	**Support** (Y/N/R)
freopen	✓	✓	Y
fscanf	✓	✓	Y
fseek	✓	✓	Y
fstat	✓	✓	Y
fsync	✓		Y
ftell	✓	✓	Y
fwrite	✓	✓	Y
getc	✓	✓	Y
getchar	✓	✓	Y
getcwd		✓	Y
gets	✓	✓	Y
getw	✓	✓	Y
ioctl	✓		Y
link	✓	✓	Y
lseek	✓	✓	Y
lstat	✓		Y
mkdir	✓		Y
mknod	✓	✓	R
mktemp	✓	✓	Y
open	✓	✓	Y
opendir	✓		Y

(continued)

Table 4-1. BSD and SysV Call Support (continued)

Function	**System**		**Support (Y/N/R)**
	BSD	**System V**	
perror	✓	✓	Y
printf	✓	✓	Y
putc	✓	✓	Y
putchar	✓	✓	Y
puts	✓	✓	Y
putw	✓		Y
read	✓	✓	Y
readdir	✓		Y
readlink	✓		Y
readv	✓		Y
rename	✓		Y
rewind	✓	✓	Y
rewinddir	✓		Y
rmdir	✓		Y
scanf	✓	✓	Y
seekdir	✓		Y
setbuf	✓	✓	Y
setbuffer	✓		Y
setlinebuf	✓		Y
setvbuf		✓	Y
sprintf	✓	✓	Y

(continued)

Table 4-1. BSD and SysV Call Support (continued)

Function	System BSD	System V	Support (Y/N/R)
sscanf	✓	✓	Y
stat	✓	✓	Y
symlink	✓		Y
telldir	✓		Y
tempnam		✓	Y
tmpfile		✓	Y
tmpnam		✓	Y
truncate	✓		Y
umask	✓	✓	Y
ungetc	✓	✓	Y
unlink	✓	✓	Y
utime	✓	✓	Y
utimes	✓		Y
vfprintf	✓	✓	Y
vprintf	✓	✓	Y
vsprintf	✓	✓	Y
write	✓	✓	Y
writev	✓		Y

mknod Restriction

The Domain NFS implementation allows the BSD and SysV **mknod** function to make directories but not special files.

4.2 Domain/OS System Calls

The Domain NFS software supports Domain/OS **name_$**, **ios_$**, and **stream_$** system calls for accessing and modifying files and directories located in foreign file systems. (You should use **ios_$** calls in new programs because these calls supersede the **stream_$** calls.) However, several functions cannot be fully supported due to the differences between Domain/OS and other operating systems. Table 4-2 lists these calls and indicates whether each call is fully supported (Y), unsupported (N), or supported with restrictions (R).

See the *Domain/OS Call Reference* for details about each call.

Table 4-2. Domain/OS Call Support

Call	**Support** (Y/N/R)
ios_$ Calls	
ios_$change_path_name	Y
ios_$close	Y
ios_$create	R
ios_$delete	Y
ios_$dup	Y
ios_$equal	Y
ios_$force_write_file	Y
ios_$get	Y
ios_$get_dir	Y
ios_$get_handle	Y
ios_$inq_byte_pos	Y
ios_$inq_conn_flags	Y

(continued)

Table 4-2. Domain/OS Call Support (continued)

Call	**Support** (Y/N/R)
ios_$ Calls (continued)	
ios_$inq_cur_rec_len	Y
ios_$inq_file_attr	Y
ios_$inq_full_key	Y
ios_$inq_mgr_flags	Y
ios_$inq_obj_flags	Y
ios_$inq_path_name	Y
ios_$inq_rec_pos	Y
ios_$inq_rec_remainder	Y
ios_$inq_rec_type	Y
ios_$inq_short_key	Y
ios_$inq_type_uid	Y
ios_$locate	Y
ios_$open	Y
ios_$put	Y
ios_$replicate	Y
ios_$seek	Y
ios_$seek_full_key	Y
ios_$seek_short_key	Y
ios_$seek_to_bof	Y
ios_$seek_to_eof	Y
ios_$set_conn_flag	R
ios_$set_dir	Y

(continued)

Table 4-2. Domain/OS Call Support (continued)

Call	**Support (Y/N/R)**
ios_$ Calls (continued)	
ios_$set_locate_buffer_size	Y
ios_$set_obj_flag	R
ios_$set_rec_type	R
ios_$switch	Y
ios_$truncate	Y
name_$ Calls	
name_$add_link	N
name_$cname	N
name_$create_directory	N
name_$create_file	N
name_$delete_directory	N
name_$delete_file	N
name_$drop_link	N
name_$extract_data	N
name_$get_ndir	Y
name_$get_path	Y
name_$get_wdir	Y
name_$read_dir	N
name_$read_link	N
name_$set_ndir	Y
name_$set_wdir	Y

(continued)

Table 4-2. Domain/OS Call Support (continued)

Call	**Support** (Y/N/R)
stream_$ Calls	
stream_$close	Y
stream_$create	Y
stream_$create_bin	N
stream_$create_here	Y
stream_$delete	Y
stream_$force_write_file	Y
stream_$get_buf	Y
stream_$get_conditional	Y
stream_$get_ec	Y
stream_$get_prior_rec	Y
stream_$get_rec	Y
stream_$inquire	Y
stream_$open	Y
stream_$put_chr	Y
stream_$put_rec	Y
stream_$redefine	R
stream_$replace	Y
stream_$seek	Y
stream_$truncate	Y

The remainder of this section describes the restrictions on each Domain/OS call that has limitations when applied to files located in foreign file systems.

ios_$create Restrictions

With **ios_$create**, you can specify any valid object type, but the Domain NFS software creates the type according to Table 4-3.

Table 4-3. Domain NFS Object Types and ios_$create

Type Requested	Type Created	Description
directory_$uid	nfs_dir_$uid	A remote directory
Any unstruct equivalent type (see Chapter 2)	nfs_file_$uid	A remote unstructured file
Any other type	None. Returns the ios_$illegal_obj_type error.	Error

ios_$set_conn_flag Restrictions

With **ios_$set_conn_flag**, only file access is allowed. Therefore, the connection flags **tty**, **ipc**, and **vt** cannot be turned on; an attempt to set these flags will result in an **ios_$illegal_operation** error.

ios_$set_obj_flag Restriction

With **ios_$set_obj_flag**, an attempt to set an object flag of **ios_$of_sparse_ok** to false is ignored.

ios_$set_rec_type Restriction

With **ios_$set_rec_type**, an attempt to set the record type to anything other than **ios_$undef** will result in an **ios_$illegal_operation** error.

stream_$redefine Restrictions

The following restrictions apply to the attributes that you can set by using the **stream_$redefine** call for a directory or file located in a foreign file system:

- Object Types
  You cannot redefine the object type.
- Sparse Flag
  You cannot set the Sparse Flag false.

4.3 Standard I/O Procedures

This section describes the Pascal I/O procedures and FORTRAN I/O statements supported by the Domain implementation of NFS. (Domain NFS software fully supports all C standard I/O calls.)

4.3.1 Domain Pascal I/O Procedures

The Domain NFS implementation fully supports the following standard predeclared Domain Pascal input/output procedures. See the *Domain Pascal Language Reference* for details about these procedures and their use.

- **close**
- **find**
- **get**
- **open**
- **page**
- **put**
- **read**
- **readln**
- **replace**
- **reset**
- **rewrite**
- **write**
- **writeln**

4.3.2 Domain FORTRAN I/O Statements

The Domain NFS implementation fully supports the following standard Domain FORTRAN input/output statements. See the *Domain FORTRAN Language Reference* for details about these procedures and their use.

- **backspace**
- **close**
- **endfile**
- **inquire**
- **open**
- **print**
- **read**
- **rewind**
- **write**

4.4 Notes for Programmers

Because of the way foreign systems manipulate files, and because of the Domain NFS implementation itself, there are several considerations that affect program performance. Keep in mind the following information when you write programs that access files located in foreign file systems:

- Byte seeks are faster than record seeks.
- The **ios_$inq_rec_pos** call requires significant internal processing to determine the end-of-file location.
- You cannot write to a directory (**nfs_dir**) or a gateway object (**nfs_gate**). However, you can create new files in the directory by using the **ios_$create** call.
- The Domain NFS implementation does not support many of the **name_$** calls. (See Table 4-2.) Use instead the **ios_$** calls indicated in Table 4-4.

Table 4-4. ios_$ Equivalents for name_$ Calls

name_$ Call	**ios_$ Call**
name_$create_file	ios_$create
name_$create_directory	ios_$create
name_$delete_file	ios_$delete
name_$delete_directory	ios_$delete
name_$read_dir	ios_dir_$readdir

Appendix A

Descriptions of Commands, Servers, and Files

This appendix contains descriptions of the following Domain NFS commands, server daemons, and files, arranged in alphabetical order (you can also access these descriptions from BSD and SysV shells via the **man** command):

exports
mount/umount (BSD)
mount/umount (SysV)
mountd
nfsd
nfsstat
portmap
rmtab
rpcinfo
showmount

exports — file specifying mountable file systems

DESCRIPTION

The **/etc/exports** file contains a list of file system names. Each file system name must appear at the beginning of a line. Following the file system name and separated by spaces or tabs are the names of the hosts authorized to remotely mount that file system. If a line contains a file system name and no host names, then any host can remotely mount the specified file system.

When **mountd** processes a mount request, it checks **/etc/exports** for the export status of the file system and checks **/etc/hosts** to validate the hosts associated with that file system. A system administrator typically maintains **/etc/exports**.

All text from a pound sign (#) to the end of the line is treated as a comment.

EXAMPLES

Following are examples of **/etc/exports** files:

```
//alewife                         #everyone

//alewife   porter                #just porter

//alewife   porter davis park     #just these three
```

FILES

/etc/exports

SEE ALSO

mountd

mount, umount (BSD) — mount and unmount file system

SYNOPSIS

Removable file systems only:

/etc/mount [**-ar**] [**-t 4.3**] [*dev dir*]

/etc/umount [**-a**] [*dev*]

Remote file systems only:

/etc/mount [**-afrvp**] [**-t nfs**] [**-o** *options*] [*fsys*] [*dir*]

/etc/umount [**-av**] [**-root**] [*fsys* | *dir*]

DESCRIPTION

The **mount** command mounts removable and remote file systems.

The **umount** command unmounts removable and remote file systems.

When mounting a *removable* file system located on a diskette, cartridge tape, or magtape, specify the device name for *dev* and a pre-existing directory pathname for *dir*.

When mounting a *remote* file system (that is, one located on another host), specify the remote host and file system in the form *host:fsname* for *fsys* and a non-existent directory for *dir*. (The **mount** command creates a gateway object named *dir*; the **umount** command deletes it.)

The **mount** and **umount** commands maintain a table of mounted file systems in **/etc/mtab**. If you execute the **mount** command without arguments, it prints the table. If you specify only one argument, the command searches **/etc/fstab** for an entry with the same argument; if the entry exists, the command then uses it to mount the file system.

OPTIONS

The command *options* can be one or more of the following items separated by a space:

–a	With **/etc/mount,** try to mount all file systems listed in the file **/etc/fstab**. (This file contains entries added manually by the user.) With **/etc/umount,** try to unmount all file systems listed in **/etc/mtab**. (This file contains an entry for every file system currently mounted by the local host.)
–f	Fake a new **/etc/mtab** entry without mounting the file system.
–o	Accept the following arguments separated by commas, a: options. The defaults are **fg, retry=1, timeo=3,** and **hard**

bg	Retry in background if server host's **mount** does not respond.
fg	Retry in foreground.
hard	Retransmit until server responds.
mtimeo=*n*	Set the mount timeout to *n* tenths of a secon
port=*n*	Set NFS port number *n*.
retrans=*n*	Set the number of NFS transmissions retri (that is, the number of attempts NFS will ma to access files in a mounted file system) to
retry=*n*	Set the number of mount retries to *n*.
ro	Allow read only.
root	Mount in the global root directory.
rsize=*n*	Set the read request buffer size to *n* bytes.
rw	Allow read/write.

soft	Report an error if server fails to respond.
timeo=*n*	Set the NFS transmission timeout to *n* tenths of a second. When attempting to access the mounted file system, NFS waits the specified amount of time for response; if there is no response, NFS multiplies *n* by 2 and retransmits the request.
wsize=*n*	Set the write request buffer size to *n* bytes.

–p	List the mounted file systems in the same format as entries in **/etc/fstab**.
–r	Mount the file system as read–only.
–root	Unmount the file system in the global root directory.
–t	Accept the following argument as the file system type; the recognized types are **4.3** for removable file systems and **nfs** for remote file systems.
–v	Display a message when mounting the file system.

FILES

/etc/mtab
/etc/fstab

BUGS

Note that physically write-protected file systems, as well as those on magnetic tape, must be mounted read-only. If not, errors develop even if no explicit write is attempted.

Mounting a removable file system that contains unreadable or otherwise invalid data can have unpredictable consequences.

SEE ALSO (in the *BSD Command Reference*)

mount(2)
mtab(5)
fstab(5)
mkdisk(8)

mount, umount (SysV) — mount and unmount file system

SYNOPSIS

Removable file systems only:

/etc/mount [**–r**] [**–f** *type*] [*dev dir*]

/etc/umount [*dev*]

Remote file systems only:

/etc/mount [**–r**] [**–f nfs**][*,options*] [*fsys*] [*dir*]

/etc/umount [**–root**] [*fsys* | *dir*]

DESCRIPTION

The **mount** command mounts removable and remote file systems.

The **umount** command unmounts removable and remote file systems.

When mounting a *removable* file system located on a diskette, cartridge tape, or magtape, specify the device name for *dev* and a pre–existing directory pathname for *dir*.

When mounting a *remote* file system (that is, one located on another host), specify the **–f** option with the argument **nfs**, the remote host and file system in the form *host:fsname* for *fsys*, and a non–existent directory for *dir*. (The **mount** command creates a gateway object named *dir*; the **umount** command deletes it.)

The **mount** and **umount** commands maintain a table of mounted file systems in **/etc/mnttab**. If you execute the mount command without arguments, it prints the table. If you specify only one argument, the command searches **/etc/fstab** for an entry with the same argument; if the entry exists, the command then uses it to mount the file system.

You must be running as root (the super–user) in order to use the **mount/umount** commands.

OPTIONS

The command *options* can be one or more of the following items separated by a space:

–f *type* Accept the argument that follows as the file system type to be mounted. If this argument is omitted, mount defaults to the root *type*. If the type is **nfs**, then accept the following arguments, separated by commas, as options. The defaults are **fg**, **retry=1**, **timeo=3**, and **hard**.

bg	Retry in background if server host's **mountd** doesn't respond.
fg	Retry in foreground.
hard	Retransmit until server responds.
mtimeo=*n*	Set the mount timeout to *n* tenths of a second.
port=*n*	Set NFS port number *n*.
retrans=*n*	Set the number of NFS transmissions retries (that is, the number of attempts NFS will make to access the mounted file system) to *n*.
retry=*n*	Set the number of mount retries to *n*.
ro	Allow read only.
root	Mount in the global root directory.
rsize=*n*	Set the read request buffer size to *n* bytes.

	rw	Allow read/write.
	soft	Report an error if server fails to respond.
	timeo=*n*	Set the NFS transmission timeout to *n* tenths of a second. When attempting to access the mounted file system, NFS waits the specified amount of time for response; if there is no response, NFS multiplies *n* by 2 and retransmits the request.
	wsize=*n*	Set the write request buffer size to *n* bytes.
–r	Mount the file system as read–only.	
–root	Unmount the file system in the global root directory.	

FILES

/etc/mnttab
/etc/fstab

BUGS

Note that physically write–protected file systems, as well as those on magnetic tape, must be mounted read–only. If not, errors develop even if no explicit write is attempted.

Mounting a removable file system that contains unreadable or otherwise invalid data can have unpredictable consequences.

SEE ALSO (in the *SysV Command Reference*)

mount(2)
mtab(5)
fstab(5)
mkdisk(8)

mountd — daemon to manage file system mount requests

SYNOPSIS

Normally started by **/etc/rc**. See description below.

DESCRIPTION

The **/etc/rc** script normally starts up **/etc/mountd**, if the file **/etc/daemons/mountd** exists. (Note that it is the existence of **/etc/daemons/mountd** that causes **etc/rc** to invoke **/etc/mountd**. The contents of the file are irrelevant.)

The **mountd** server daemon runs on a server host and manages file system mount requests. It must be run by root. When **mountd** receives a request from a client host, it consults the server host's **/etc/exports** file to determine if the client host can access the requested file system. The daemon also supplies the **showmount** command with information about client hosts that have file systems mounted.

The **portmap** daemon must be running on a host before **mountd** is invoked on that host.

SEE ALSO

nfsd
portmap

nfsd — daemon to service file system mount requests

SYNOPSIS

Normally started by **/etc/rc**. See description below.

DESCRIPTION

The **/etc/rc** script normally starts up **/etc/nfsd**, if the file **/etc/daemons/nfsd** exists. (Note that it is the existence of **/etc/daemons/nfsd** that causes **etc/rc** to invoke **/etc/mountd**. The contents of the file are irrelevant.)

The **nfsd** server daemon runs on a server host and services the file system mount requests of client hosts. It must be run by root.

The **portmap** and **mountd** daemons must be running on a host before **nfsd** is invoked on that host.

SEE ALSO

mountd
portmap

nfsstat — display Network File System statistics

SYNOPSIS

/etc/nfsstat [**-csnrz**]

DESCRIPTION

The **nfsstat** command displays statistical data about NFS; you can also use it to reinitialize the statistics. Without options, the command defaults to **nfsstat -csnr**.

OPTIONS

-c	Display client statistics. You can combine this option with the **-n** and **-r** options to print only client NFS or client RPC (Remote Procedure Call) statistics.
-s	Display server statistics in the same manner as the **-c** option.
-n	Display NFS statistics for both client and server. You can combine this option with the **-c** and **-s** options to print only client or server NFS statistics.
-r	Display RPC statistics in the same manner as the **-n** option.
-z	Zero (reinitialize) statistics. You can combine this option with any of the other options to reinitialize particular sets of statistics after printing them. This option does not require super-user privileges.

portmap — daemon to map RPC (Remote Procedure Call) program numbers to DARPA protocol port numbers

SYNOPSIS

Normally started by **/etc/rc**. See description below.

DESCRIPTION

The **/etc/rc** script normally starts up **/etc/portmap**, if the file **/etc/daemons/portmap** exists. (Note that it is the existence of **/etc/daemons/portmap** that causes **/etc/rc** to invoke **/etc/portmap**. The contents of the file are irrelevant.)

The **portmap** server daemon runs on a server host and directs data transmissions from client hosts to RPC servers. When an RPC server begins operation, it tells **portmap** the port number being monitored and the program numbers being serviced by the RPC server. A client host sending RPC packets to a particular program number gets the appropriate port number from **portmap**.

SEE ALSO

mountd
nfsd

rmtab — table of file systems mounted by remote hosts

DESCRIPTION

The **/etc/rmtab** file contains information about file systems that remote hosts have mounted. It is an internal file used by the **showmount** command; users never reference it directly.

FILES

/etc/rmtab

SEE ALSO

showmount

rpcinfo — report RPC (Remote Procedure Call) information

SYNOPSIS

etc/rpcinfo -p *host*

/etc/rpcinfo -u *host program version*

DESCRIPTION

The **rpcinfo** command makes an RPC call to an RPC server and reports the results.

OPTIONS

-p Probe the portmapper on *host* and print a list of all registered RPC programs.

-u Using UDP, make an RPC call to procedure 0 of the specified *version* of *program* on the specified *host*, and report if a response is received.

showmount — show all remote mounts

SYNOPSIS

/etc/showmount [**-a**] [**-d**] [**-e**] *host*

DESCRIPTION

The **showmount** command lists all the clients that have remotely mounted a file system belonging to *host*. The command gets its information from the **mountd** server on *host*.

OPTIONS

-d	List directories that have been remotely mounted by clients.
-a	Print all remote mounts in the format *hostname:directory*, where *hostname* is the name of the client and *directory* is the root of the mounted file system.
-e	Print the list of exported file systems in **/etc/exports**.

FILES

/etc/fstab	file system table
/etc/mtab	mount table(BSD)
/etc/mnttab	mount table (SysV)
/etc/rmtab	remote mount table

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes **/etc/umount**.

SEE ALSO

mountd
exports
rmtab

Appendix B

Managing NFS on the Domain Network

Before installing Domain NFS software, you need to determine the optimal server arrangement for your site and you need to establish uniform user and group IDs on all NFS hosts. These two tasks are described in Sections B.1 and B.2.

After you have installed Domain NFS software, you must ensure that the IDs for any newly added users and groups are also uniform. Section B.3 describes procedures for adding new users and groups.

B.1 Arranging Servers on the Domain Network

With most NFS implementations, a user intending to access several remote file systems must individually mount each file system. In addition, the hosts on which the remote file systems reside must be server hosts. In Figure B–1, users at hosts **fs1**, **fs2**, and **fs3** can access each other's file systems because all three hosts are server hosts.

With Domain NFS software, you could similarly make every Domain workstation a server host. Doing so would make all remote file systems on the network available to users at hosts **fs1**, **fs2**, and **fs3** — and vice versa. Figure B-1 illustrates this arrangement when Domain workstations are connected to foreign systems through a gateway node. Figure B-2 illustrates this arrangement when Domain workstations and foreign systems are attached to the same network.

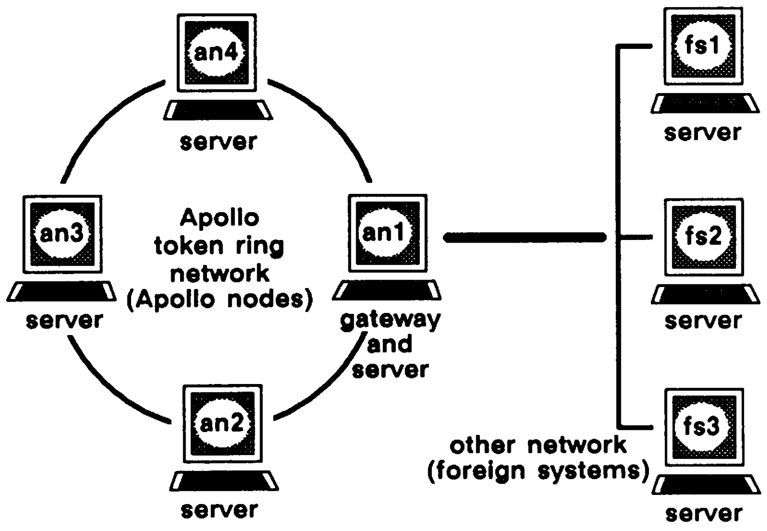

Figure B-1. Sample Server Arrangement in a Gateway Configuration

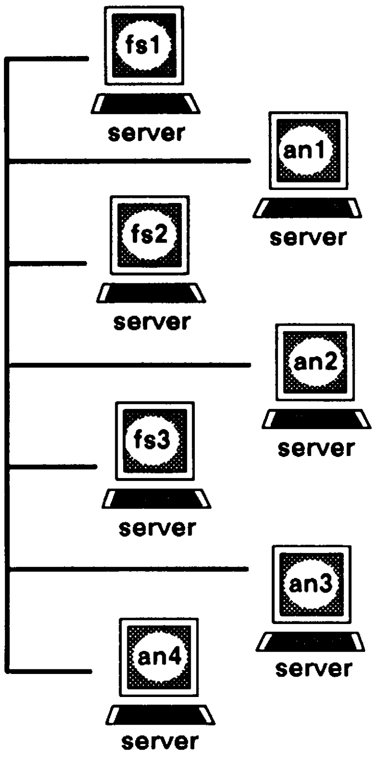

Figure B–2. Sample Server Arrangement in a Direct–Connect Configuration

In either of the configurations shown in Figure B–1 and Figure B–2, a user at host **fs1** could use the following mount commands to mount the remote slash (/) file systems on both Domain and foreign hosts. Note that the user would probably need many more mount commands to mount all file systems on **fs2** and **fs3**.

```
$ /etc/mount -o soft fs2:/ /fs2_root                (BSD)
$ /etc/mount -o soft fs3:/ /fs3_root
$ /etc/mount -o soft an1:/ /an1_root
$ /etc/mount -o soft an2:/ /an2_root
$ /etc/mount -o soft an3:/ /an3_root
$ /etc/mount -o soft an4:/ /an4_root
$ /etc/mount -f nfs,soft fs2:/ /fs2_root          (SysV)
$ /etc/mount -f nfs,soft fs3:/ /fs3_root
$ /etc/mount -f nfs soft an1:/ /an1_root
$ /etc/mount -f nfs,soft an2:/ /an2_root
$ /etc/mount -f nfs,soft an3:/ /an3_root
$ /etc/mount -f nfs,soft an4:/ /an4_root
```

Users at hosts **fs2** and **fs3** would also need to mount the remote root (/) file systems in both networks. Note that users on all three foreign systems would have to create and/or set aside one directory for every mount command executed (a requirement of other NFS implementations).

The scenario we have described thus far is what users at hosts **fs1**, **fs2**, and **fs3** *could* do if all Domain hosts were server hosts. However, what they *need* to do in this situation is actually much easier, as you will see in the next subsection.

B.1.1 Single Domain NFS Server

The Domain/OS network root directory (//) allows a user at a Domain workstation to access directories and files on all Domain workstations — without requiring the user to explicitly mount file systems. Domain NFS software offers a similar capability to users at foreign hosts with the single server configuration.

Figure B–3 and Figure B–4 illustrate the single Domain NFS server configuration. With this configuration, a user at a foreign host can access file systems on any Domain workstation by mounting the network root directory through the Domain server host.

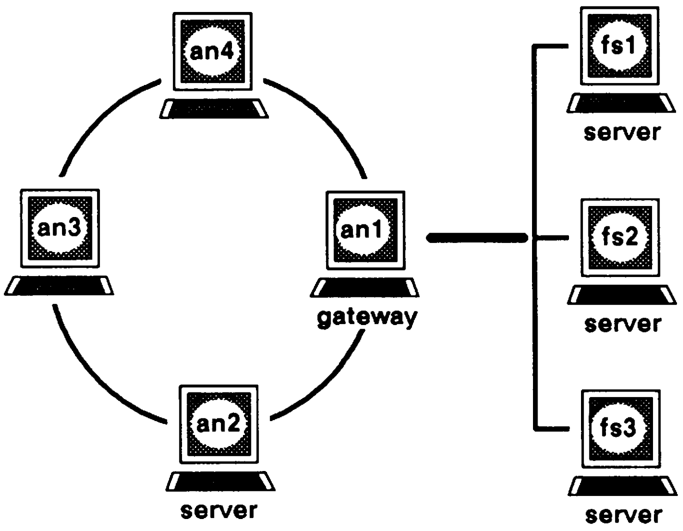

Figure B–3. Gateway Configuration with a Single Domain NFS Server

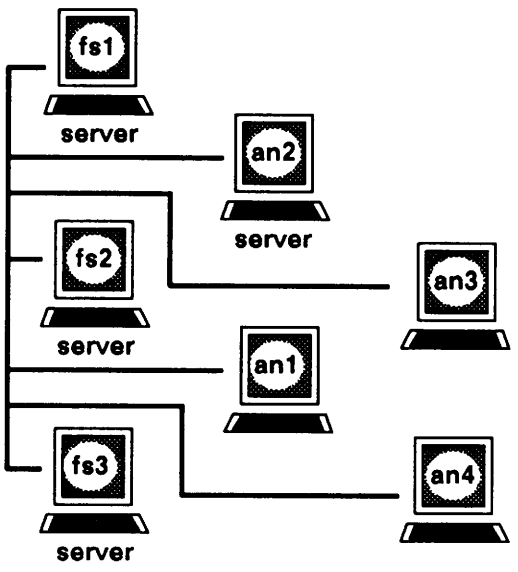

Figure B-4. Direct-Connect Configuration with a Single Domain NFS Server

For example, a user at host **fs1** can access the root (/) file systems of all hosts on the networks in Figure B-3 and Figure B-4 after issuing the following mount commands:

```
$ /etc/mount -o soft fs2:/ /fs2_root                (BSD)
$ /etc/mount -o soft fs3:/ /fs3_root
$ /etc/mount -o soft an2:// /apollo

$ /etc/mount -f nfs,soft fs2:/ /fs2_root            (SysV)
$ /etc/mount -f nfs,soft fs3:/ /fs3_root
$ /etc/mount -f nfs,soft an2:// /apollo
```

As in the previous example, the first two commands explicitly mount the root (/) file systems located on the foreign hosts **fs2** and **fs3**. However, the third command now provides access to all Domain workstations. In particular, the command mounts the network root directory (//) as **/apollo** through the server host **an2**, thereby enabling the user at host **fs1** to access a Domain object by using **/apollo** as a prefix to the object's pathname. For example, the user can now access the file **//an3/test** with the pathname **/apollo/an3/test**. This approach significantly reduces the time, effort, and the number of directories needed by other NFS implementations to mount a large number of Domain file systems.

The network root directory also simplifies the mount procedure for Domain NFS users wanting to mount remote file systems located on foreign hosts. Once a Domain NFS user mounts a remote file system in the network root directory, all Domain NFS users can access that file system by using the name specified in the mount command — without having to issue their own mount commands.

For example, suppose the user at host **an1** issues the following mount command:

```
$  /etc/mount -o soft,root fs2:/usr //fs2_usr            (BSD)

$  /etc/mount -f nfs,soft,root fs2:/usr //fs2_usr        (SysV)
```

Thereafter, any Domain NFS user who wants to access the **/usr** file system on **fs2** only needs to use the pathname **//fs2_usr**. Note that the Domain NFS mount command must include the **root** option to mount a file system in the root directory.

B.1.2 Multiple Domain NFS Servers

A single Domain NFS server host is sufficient for many sites. However, your site may require multiple Domain NFS server hosts to efficiently handle the needs of users at foreign hosts. The optimum ratio of users to server hosts varies with each site.

When you have multiple Domain server hosts, you should divide all users requiring remote access to Domain file systems into groups — with one group for each server host. All users in the same group should then mount the network root directory through the same server host.

For example, Figure B–5 and Figure B–6 illustrate sites with multiple server hosts. If the users at foreign hosts **fs1** and **fs2** mount the network root directory through the Domain NFS server host **an4** with the command

```
$ /etc/mount -o soft an4:// /apollo          (BSD)

$ /etc/mount -f nfs,soft an4:// /apollo      (SysV)
```

and the user at foreign host **fs3** uses host **an2**

```
$ /etc/mount -o soft an2:// /apollo          (BSD)

$ /etc/mount -f nfs,soft an2:// /apollo      (SysV)
```

then all three users can access Domain file systems by using pathnames beginning with **/apollo** — even though the users mounted the network root directory at different server hosts.

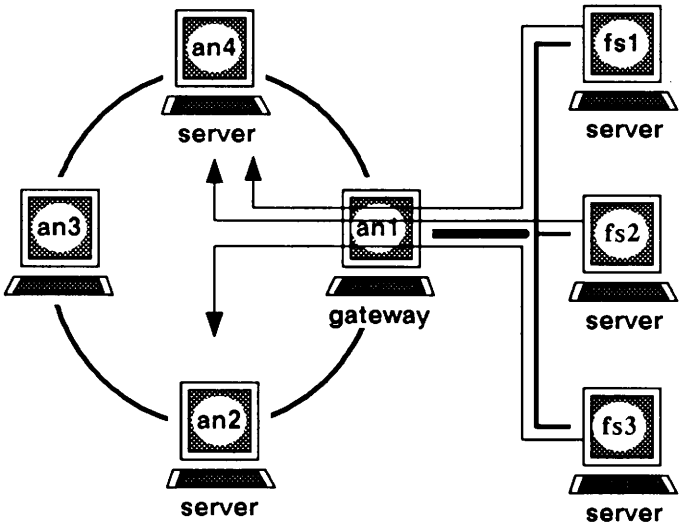

Figure B–5. Gateway Configuration with Multiple Domain NFS Servers

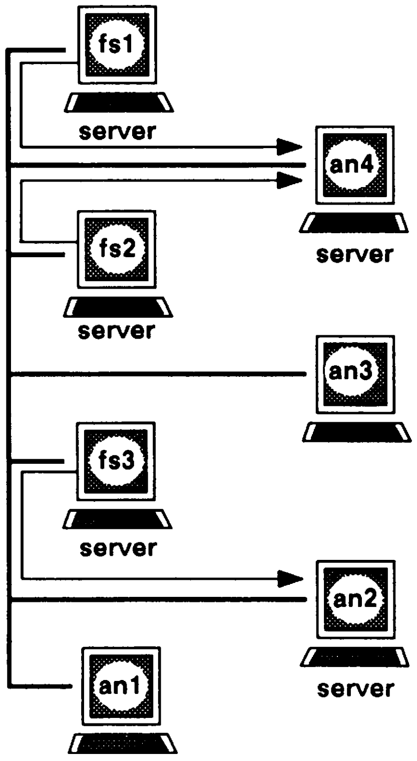

Figure B–6. Direct–Connect Configuration with Multiple Domain NFS Servers

B.2 Establishing Uniform User and Group IDs

For correct checking of access rights via NFS, you must ensure that each NFS user at your site has the same user and group IDs on all of the NFS hosts. NFS authentication is based on the numerical IDs of users and groups, not their names. The IDs for all users and all groups must be uniform on all NFS hosts.

We strongly recommend that you establish uniform user and group IDs before you install NFS. You must establish uniform IDs before using NFS.

Domain/OS provides a utility called **/etc/import_passwd** that can help you to identify and resolve possible conflicts of names and IDs. Another utility called **/etc/syncids** fixes the user and group IDs stored as part of the protection information for each file and directory on an Apollo file system.

For detailed information about **import_passwd**, you should refer to the *Managing System Software* manual for your environment. Here, we give general guidelines for using **import_passwd** and **syncids**.

The **import_passwd** utility imports user and group information from a foreign system's **passwd** and **group** files to the Domain/OS network registry. It detects and attempts to resolve any conflicts between the UNIX IDs used in the foreign files and those used in the Domain/OS registry.

When you invoke **import_passwd**, you choose between two modes for resolving conflicts: one that favors information from the foreign system and one that favors information from the Apollo registry. In the "favor-foreign" mode, **import_passwd** retains IDs from the foreign system whenever possible and changes IDs in the Apollo registry to match the foreign ones. In the "favor-Apollo" mode, **import_passwd** retains IDs from the Apollo registry; to resolve conflicts, you must change IDs on the foreign systems.

Regardless of which mode you choose, you must ensure that the UNIX user and group IDs stored in your file systems match the IDs that are stored in the Apollo registry and in the foreign password and group files.

- If IDs were changed in the Apollo registry, you must run the **syncids** utility on every Apollo node to update the UNIX IDs in the Apollo file systems. (On nodes that have more than one disk volume, you must run **syncids** on each volume.) If your Apollo registry is replicated, then before you run **syncids**, you should use the **lrep -state** command in the **/etc/rgy_admin** tool to verify that all slave registry servers are up to date.

- If IDs were changed on the foreign systems, you must use the **chown** and **chgrp** utilities to update the UNIX user and group IDs on all affected files and directories in the foreign file systems.

At most sites, it is easier to run **syncids** on every Apollo node than to run **chown** and **chgrp** on every affected file in every foreign file system, so we recommend running **import_passwd** in favor–foreign mode.

Some names and IDs in the Apollo registry are **reserved** and cannot be changed, even in favor–foreign mode. If a conflict involves reserved registry entries, you must resolve the conflict on the foreign system.

At sites that have many Apollo systems and few foreign UNIX systems, it may be easier to run **import_passwd** in favor–Apollo mode. In this mode, you resolve all conflicts by changing information on the foreign system. For any names and IDs that change, you will have to use **chown** or **chgrp** to change the ownership for any affected objects in the foreign file systems. Potentially, this process must be repeated throughout every file system, so you should use favor–Apollo mode only if you expect a small number of changes on a small number of foreign systems. The **find** utility can be useful for changing ownerships throughout a file system.

Reference documentation for **import_passwd**, **rgy_admin**, and **syncids** is available online and in the *Command Reference* manual for your operating environment.

B.3 Adding New Users and Groups

Once you have established uniform IDs for the existing users and groups at your site, you must ensure that the IDs of any subsequently added users and groups also have uniform IDs on all of your NFS hosts.

When you add a new user at your site, you should first add the user to the Domain/OS network registry. (Refer to the *Managing System Software* manual for your environment for directions.) Then add the new user to the **/etc/passwd** files on your foreign systems, assigning to the user the same user and group IDs as those used in the Domain/OS network registry.

Similarly, when you add a new group at your site, you should first add the group to the Domain/OS network registry, then add the group to the **/etc/group** files on your foreign hosts, assigning the same group ID as the one used in the Domain/OS network registry.

Appendix C

Troubleshooting Guide

This appendix describes problems that may result from improper installation or use of the NFS software and provides solutions whenever possible. Section C.1 describes problems that you may encounter while at a Domain host. Section C.2 describes problems that you may encounter while at a foreign host.

C.1 Possible Problems at a Domain Host

Cannot mount a remote file system

You're unable to mount a remote file system.

Check the **/etc/exports** file on the host where the file system resides. If the file does not give you access to the file system, then either edit the file or ask someone to do it for you.

Another possibility is that the foreign host where the file system resides isn't running the NFS server. If so, start the following server daemons on the host in the specified order: **/etc/portmap**, **/etc/mountd**, and **/etc/nfsd**.

Cannot mount a remote file system from a gateway node

When you attempt to mount a remote filesystem from a gateway node, you get an error message that reads "mount: not in export list for *host*:*dir*."

Check the **/etc/exports** file on the host where the file system resides. If the file gives the gateway node access to the file system, then check the **/etc/hosts** file on the Domain network. The file should list the gateway node's address for the foreign network first and the gateway node's Domain address second.

Directory appears to be empty

When you attempt to access a remote directory through a mounted file system, the directory appears to be empty, though you know it's not.

The remote directory may actually be in a different file system. Try mounting it via the **/etc/mount** command.

C.2 Possible Problems at a Foreign Host

Cannot access Domain objects, but others can

You cannot access objects on the Domain network from a foreign system, even though others can access the same Domain objects from the same foreign system.

Compare the **/etc/passwd** files on the foreign system and the Domain network. If you have different user and group IDs in the two files, ask your system administrator to resolve the differences.

Cannot access some DSEE objects

You cannot access DSEE casehm files on the Domain network with extended pathnames of the form *//node/dir/file/version_number*.

The NFS software does not access such objects; see Chapter 2 for details. To access such objects from a foreign system, you need to log in to a Domain workstation via the **telnet** or **rlogin** command.

Cannot mount a remote file system

You're unable to mount a remote file system.

Check the **/etc/exports** file on the host where the file system resides. If the file doesn't give you access to the file system, then either edit the file or ask someone to do it for you.

Another possibility is that the Domain host used to mount the Domain file system isn't running the NFS server. If so, start the following server daemons on the host in the specified order: **/etc/portmap**, **/etc/mountd**, and **/etc/nfsd**.

Domain file briefly stays locked after NFS access

The NFS server waits 3 to 6 seconds after accessing a Domain file to unlock it. This allows the NFS server to reduce the number of times it opens and closes the same file.

ls -l of // runs very slowly or does not run properly	We suggest that you avoid using the **ls -l** command to list the contents of a mount point that is directly connected to the Domain network root directory. See Chapter 2 for more information.

Appendix D

Error Messages

This appendix lists in alphabetical order the error messages issued by the **/etc/mount** and **/etc/umount** commands.

Note that the *italicized text* simply indicates the type of information that appears in an error message. You should therefore consider only the **boldfaced text** when looking for a message in this appendix.

mount: directory path '*pathname*' must begin with '/'

You invoked the **/etc/mount** command and specified an invalid *pathname* as the mount point. The pathname of a mount point must begin with a slash (/) as in **//chauncy/mntpt** or **/mntpt**.

mount: gateway object *pathname* already exists

The **/etc/mount** command could not create the gateway object or mount point *pathname* because it already exists. If the gateway object or mount point is a directory, the following message also appears:

Domain mounts cannot be performed on an existing object. The gateway object is a file created by the mount command.

Domain NFS software creates a gateway object (a special file) when you mount a remote file system; most other NFS implementations use an existing directory.

mount: *host:fs* on *pathname*: No such file or directory

You attempted to mount the remote file system *host:fs* at the mount point *pathname* with the **/etc/mount** command. The file system *fs* does not exist on the remote host *host*.

mount: *host:fs* server not responding: RPC_TIMED_OUT

You attempted to mount the remote file system *host:fs* with the **/etc/mount** command. However, the NFS server responsible for that file system didn't respond, so the command aborted.

mount: *name* is an improper NFS FileSystem name

You invoked the **/etc/mount** command and specified the unknown file system *name*. The NFS protocol requires file system names in the format *host:filesystem*.

mount: '*options*' is an invalid option list

You invoked the **/etc/mount** command and specified the unrecognized *options*.

mount: ***pathname*** **already exists**
mount: ***pathname*** **already exists in root**

You tried to create the mount point *pathname* in the network root directory by using the **/etc/mount** command with the **root** option; however, *pathname* already exists. The **mount** command requires a non–existent pathname.

mount: ***pathname*** **is not a root**

You invoked the **/etc/mount** command with the **root** option and specified the invalid *pathname* as the mount point. A mount point specified with the **root** option must begin with two slashes (//).

mount: not in export list for ***host:fs***

Your local host (the system that printed this message) does not have access rights to the file system *host:fs*. The **/etc/exports** file on the server host *host* determines which client hosts can access which of its file systems.

mount: rpc error ***"message"***

The RPC (Remote Procedure Call) error described by *message* occurred during your attempt to mount a remote file system with the **/etc/mount** command.

mount: unable to create ***pathname*** **in local root**

The **/etc/mount** command could not create the mount point entry *pathname* in the local copy of the network root directory. Check the protection of // and try the command again.

mount: unable to create gateway object ***name***

The **/etc/mount** command could not create the gateway object *name* in the **'node_data** directory. Check the protection of **'node_data** and try the command again.

mount: unknown host *name*

You invoked the **/etc/mount** command and specified the unknown host *name*. The file **/etc/hosts** lists all valid hosts.

***pathname* not mounted**

You invoked the **/etc/umount** command and specified a file system or mount point not found in the host's **/etc/mtab** file.

umount: directory path '*pathname*' must begin with '/'

You invoked the **/etc/umount** command and specified an invalid *pathname*. The pathname of a mount point must begin with a slash (/) as in **//chauncy/mntpt** or **/mntpt**.

umount: must specify –root for unmounting *pathname*

You didn't use the **–root** option with the **/etc/umount** command to unmount a file system mounted as *pathname* in the network root directory.

umount: No entry in mtab file for *pathname1*. Deleting gateway object "*pathname2*"

You invoked the **/etc/umount** command with the **–root** option to unmount a file system mounted as *pathname1* in the network root directory. Though the local host does contain the gateway object *pathname2* that corresponds to the mounted file system, the host's **/etc/mtab** file does not contain an entry for it. The command deleted the gateway object anyway.

umount: *pathname* not in global root. Unmounting anyway

You unmounted a file system mounted in the network root directory, even though the file system did not have the specified mount point *pathname*. However, the mounted file system did have the correct **/etc/mtab** entry and gateway object before you deleted them with the **/etc/umount** command.

Sometimes mount points disappear from the network root directory because the system administrator has reset the naming server's database to remove obsolete entries. Since mount points do not respond to naming server requests, the naming server deletes mount point entries when rebuilding its database.

umount: umount –root *pathname* must be made from node *//nodename*

You attempted to unmount a file system mounted as *pathname* in the network root directory from the wrong node. (You can unmount a file system only from the same node that mounted it.) Execute the **/etc/umount** command with the **–root** option on the node *//nodename*.

Index

Reader's Response

Please take a few minutes to send us information we can use to revise and improve our manuals.

Document Title: *Using NFS on the Domain Network*
Order No.: 010414-A00
Date of Publication: June, 1988

What type of user are you?

____ System programmer; language ________________________
____ Applications programmer; language ________________________
____ System maintenance person

____ System Administrator	____ Student
____ Manager/Professional	____ Novice
____ Technical Professional	____ Other

How often do you use the Domain system? ________________________

What additional information would you like the manual to include? _____

Please list any errors, omissions, or problem areas in the manual by page, section, figure, etc. ________________________

Your Name Date

Organization

Street Address

City State Zip

No postage necessary if mailed in the U.S.

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

Reader's Response

Please take a few minutes to send us information we can use to revise and improve our manuals.

Document Title: *Using NFS on the Domain Network*
Order No.: 010414-A00
Date of Publication: June, 1988

What type of user are you?

____ System programmer; language ____________________

____ Applications programmer; language ____________________

____ System maintenance person

____ System Administrator	____ Student
____ Manager/Professional	____ Novice
____ Technical Professional	____ Other

How often do you use the Domain system? ____________________

What additional information would you like the manual to include? _____

Please list any errors, omissions, or problem areas in the manual by page, section, figure, etc. ____________________

Your Name Date

Organization

Street Address

City State Zip

No postage necessary if mailed in the U.S.

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