

**ULTRIX-32  
Guide to the  
Network File System**

Order No. AA-ME99A-TE

---

ULTRIX-32 Operating System, Version 3.0

Digital Equipment Corporation


Copyright © 1987, 1988 Digital Equipment Corporation  
All Rights Reserved.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

DEC	Q-bus	VAX
DECnet	RT	VAXstation
DECUS	ULTRIX	VMS
MASSBUS	ULTRIX-11	VT
MicroVAX	ULTRIX-32	ULTRIX Worksystem Software
PDP	UNIBUS	

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

IBM is a registered trademark of International Business Machines Corporation.

MICOM is a registered trademark of Micom System, Inc.

This manual was written and produced by the ULTRIX Documentation Group in Nashua, New Hampshire.

# Contents

## About This Guide

Audience .....	vii
Organization .....	vii
Related Documents .....	viii
Conventions .....	viii

## 1 Introduction to the Network File System

1.1 The Client and Server Relationship .....	1-2
1.2 The NFS Service .....	1-3
1.3 The NFS Locking Service .....	1-4

## 2 Setting Up the Network File System

2.1 Preparing to Set Up NFS .....	2-1
2.2 Setting Up NFS Automatically .....	2-3
2.3 Setting Up NFS Manually .....	2-4
2.3.1 Edit the /etc/exports File .....	2-4
2.3.2 Edit the /etc/rc.local File .....	2-5
2.3.2.1 Make an Entry for the portmap Daemon .....	2-5
2.3.2.2 Make an Entry for the mountd Daemon .....	2-5
2.3.2.3 Make Entries for the NFS Daemons .....	2-6

2.3.2.4 Make an Entry for the NFS Locking Service .....	2-6
2.3.3 Edit the /etc/fstab File .....	2-7
2.3.4 Reboot the System .....	2-7
2.4 Modifying the NFS Environment .....	2-8
2.4.1 Modifying the NFS Environment Automatically .....	2-8
2.4.2 Modifying the NFS Environment Manually .....	2-9

### **3 Managing the Network File System**

3.1 Mounting a Remote File System or Directory .....	3-1
3.2 Programming With the NFS Locking Service .....	3-3
3.3 Locking Remote Files .....	3-4
3.4 Recovering NFS locks .....	3-5
3.5 Addressing System Security .....	3-6
3.5.1 Superuser Access over the Network .....	3-6
3.5.2 Mailing to Superuser (root) Across NFS Systems .....	3-7
3.5.3 Port Monitoring .....	3-9
3.5.4 Increasing System Security .....	3-10
3.6 Identifying System Differences .....	3-11
3.6.1 Some File Operations Fail .....	3-12
3.6.2 System (Clock) Times Can Vary .....	3-12

### **4 Troubleshooting the Network File System**

4.1 Network or Server Failures .....	4-1
4.2 Remote Mount Failures .....	4-3
4.3 Remote Mount Error Messages .....	4-4
4.4 Client Programs Block .....	4-7
4.4.1 The NFS Server is Down .....	4-7

4.4.2 The nfsd Daemon is Malfunctioning .....	4-8
4.4.3 Two Or More Processes Are Deadlocked .....	4-8
4.5 System Hangs Part Way Through Boot .....	4-8
4.6 Slow Remote File Access .....	4-9
4.7 NFS Console Error Messages .....	4-10

## Index

## Figures

1-1: Host Systems Running NFS .....	1-2
3-1: Relationship of Client and Server with the NFS Locking Service .....	3-5



## About This Guide

The objective of this guide is to provide introductory, setup, and troubleshooting information for the Network File System (NFS). This guide will also assist you in developing NFS management procedures. It presents guidelines from which you can develop specific procedures for your site.

### Audience

This guide is meant for the person responsible for maintaining networks on an ULTRIX operating system. This person is usually the system manager, but could be a network manager or the system manager who is also a user of a MicroVAX processor running the ULTRIX operating system. This guide assumes that the reader is familiar with the ULTRIX system commands, the system configuration, the naming conventions, and an editor such as vi or ed. It also assumes the reader knows the names and addresses of the other systems on the network.

### Organization

This guide consists of four chapters and an index. The chapters are:

#### Chapter 1: Introduction

This chapter introduces the Network File System (NFS) and describes basic NFS concepts. It provides the background information needed before you can set up and run NFS on your system. Basic NFS concepts are described.

#### Chapter 2: Setting Up the Network File System

This chapter describes how to set up NFS automatically on your system, using the nfssetup command. It also describes how to set up NFS manually. The manual description is included for those who want to understand how NFS operates and which files are affected by NFS.

#### Chapter 3: Managing the Network File System

This chapter describes how to mount and unmount remote file systems over NFS and provides general considerations, such as the implications of mounting a remote file system on a local mount point.

This chapter also discusses the optimum NFS configurations, such as how many NFS daemons should be running.

This chapter addresses system security with NFS and provides information on how to improve security.

Finally, this chapter addresses system behaviors that you may notice while running NFS on your system that were not apparent before NFS was set up.

#### Chapter 4: Troubleshooting NFS

This chapter describes the basic approach to solving NFS-related system problems. It discusses various system problems you may encounter and describes how to solve them. In addition, this chapter lists the NFS error messages and suggested user actions.

## Related Documents

You should have available the related hardware documentation for your system. You also should have the other documents in the ULTRIX documentation set, including the ULTRIX Reference Pages.

## Conventions

The following conventions are used in this guide:

special	In text, each mention of a specific command, option, partition, pathname, directory, or file is presented in this type.
command(x)	In text, cross-references to the command documentation include the section number in the reference manual where the commands are documented. For example: See the <code>cat(1)</code> command. This indicates that you can find the material on the <code>cat</code> command in Section 1 of the ULTRIX Reference Pages.
literal	In syntax descriptions, this type indicates terms that are constant and must be typed just as they are presented.
<i>italics</i>	In syntax descriptions, this type indicates terms that are variable.
[ ]	In syntax descriptions, square brackets indicate terms that are optional.
. . .	In syntax descriptions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.



<code>function</code>	In function definitions, the function itself is shown in this type. The function arguments are shown in italics.
<code>UPPERCASE</code>	The ULTRIX system differentiates between lowercase and uppercase characters. Enter uppercase characters only where specifically indicated by an example or a syntax line.
<code>example</code>	In examples, computer output text is printed in this type.
<b><code>example</code></b>	In examples, user input is printed in this bold type.
<code>%</code>	This is the default user prompt in multiuser mode.
<code>#</code>	This is the default superuser prompt.
<code>&gt;&gt;&gt;</code>	This is the console subsystem prompt.
<code>.</code>	In examples, a vertical ellipsis indicates that not all of the lines of the example are shown.
<code>.</code>	
<code>.</code>	
<code>KEYNAME</code>	In examples, a word or abbreviation in angle brackets indicates that you must press the named key on the terminal keyboard.
<code>CTRL/x</code>	In examples, symbols like this indicate that you must hold down the CTRL key while you type the key that follows the slash. Use of this combination of keys may appear on your terminal screen as the letter preceded by the circumflex character. In some instances, it may not appear at all.



# Introduction to the Network File System 1

This chapter provides an overview of the Network File System (NFS), discusses how to maintain NFS, and describes the related files.

NFS is a facility for sharing files in a heterogeneous environment of processors, operating systems, and networks. Sharing is accomplished by mounting a remote file system or directory and then reading or writing the files as though they were local. This sharing removes the need to copy files across the network from one system to another and makes it easier to access remote files. It also reduces the risk of having copies of the same file on different systems become out-of-date, because the sole copy can be kept on a single host system.

Client systems request resources provided by other systems, called servers. A client is any host system or process generated by a piece of software, such as NFS or Yellow Pages (YP), that uses services from a server. YP is discussed in the Guide to the Yellow Pages Service.

A server is any host system or process that provides a network service such as NFS or YP. A single host can provide more than one service. Servers are passive; they wait for clients to call them. Servers never call the clients.

There is not necessarily a one-to-one correspondence between servers, clients, and systems. A system that acts as a server can also act as a client. For example, NFS allows clients to mount remote file systems and directories and to access the files within the remote directories locally, provided a server has exported the directory. A server that exports file systems and directories can also mount remote file systems and directories exported by other systems, thus becoming a client.

To export a file system or directory is to make it available for NFS clients to mount remotely onto their local systems. The `/etc/exports` file defines the NFS file systems and directories that can be exported.

A server makes specified file systems and directories available which clients can then mount as remote file systems and directories. In this way, users can access remote files as if they were on the local system.

The client always initiates the remote mount. The server completes the binding subject to access control rules specific to NFS. Because most

network administration problems occur at bind time, you should know how a client binds to a server and what access control policy each server uses. If you are running NFS, you may elect to use the NFS locking service. This service supports file and file region advisory locking on local and remote systems. This is important when several users or processes access the same file simultaneously.

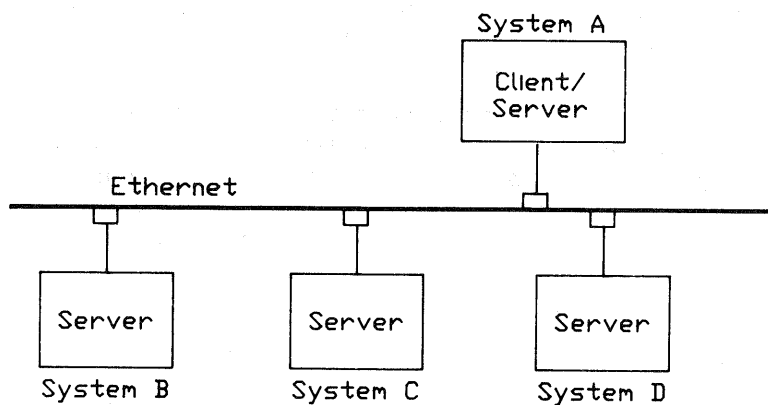
The topics discussed in this chapter are:

- The client and server relationship
- The NFS service
- The NFS locking service

## 1.1 The Client and Server Relationship

An NFS client selects a specific server from which to mount a given file system or directory. It can choose to mount file systems and directories from a number of servers.

Figure 1-1 shows the relationship between several host systems connected by an Ethernet cable to the same network. There is one NFS client and four NFS servers on the network shown. Note that System A is both an NFS client and server.



ZK-0011U-GE

**Figure 1-1: Host Systems Running NFS**

## 1.2 The NFS Service

This section briefly describes the service that NFS provides. It describes the `/etc/fstab` and `/etc/exports` files and identifies the four programs that implement NFS.

NFS enables users to share files over the network. A client can mount or unmount file systems and directories from an NFS server. The client always initiates the binding to a server's file system or directory by using the `mount` command. Once a remote file system or directory is mounted, it is treated as a file system by the client. Typically, a client mounts one or more remote file systems and directories at startup time, if lines similar to the following are in the `/etc/fstab` file. The `mount` command reads these lines when the system comes up:

```
/usr2@titan:/usr2:rw:0:0:nfs:hard,bg:
/usr/man@venus:/usr/man:rw:0:0:nfs:hard,bg:
```

See `fstab(5)` in the ULTRIX Reference Pages for a description of the file format.

NFS servers control who can mount their resources by limiting named file systems and directories to a specific set of clients with an entry in the `/etc/exports` file. Note that the `/etc/exports` file allows you to limit access to NFS clients, but not to individual remote users.

Each NFS request is checked to make sure the file system or directory being accessed is currently exported. If it is not, the operation fails. The `mountd` daemon marks exported file systems and directories based upon the information in the `/etc/exports` file. If this file is modified, the next time the `mountd` daemon processes a request, it updates the export state on the server.

The `/etc/exports` file defines which file systems and directories can be exported to which hosts on the network. There can be only one entry in the `/etc/exports` file per file system or directory being exported. The following example shows entries for the `/etc/exports` file:

```
/usr/staff/doe green      # export directory only to host green
/usr/staff -o pink        # export file system read only to host pink
/usr/local                # export file system to the world
/usr2 pink green black    # export file system only to these hosts
```

In the example above, the directory `/usr/staff/doe` is exported to the host system named `green`. The entire file system `/usr/staff`, including the directory `/usr/staff/doe`, is exported with read permissions, only, to the host system named `pink`. The file system `/usr/local` is exported to all the systems connected to the local network and running NFS. The file system

/usr2 is exported to the host systems named pink, green, and black.

### **Note**

The pathnames specified in the /etc/exports file can be any local directory and are not limited to file system mount points. However, exports do not cross file system boundaries. For example, if / and /usr are separate file systems on the server, then exporting the file system / does not allow NFS clients to access files in the /usr file system. See `exports(5nfs)` in the ULTRIX Reference Pages for a description of the additional options and file format.

Four programs implement the NFS service: `portmap`, `mountd`, `biod`, and `nfsd`. A client's mount request is transmitted to the remote server's `mountd` daemon after obtaining its address from `portmap`. A port mapper is a Remote Procedure Call (RPC) daemon that maps RPC program numbers of network services to their User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) protocol port numbers.

The `mountd` daemon checks the access permission of the client and returns a pointer to the file system or directory. After the mount completes, access to that mount point and below goes through the pointer to the server's NFS daemon (`nfsd`) using remote procedure calls (`rpc`). Some file access requests (write-behind and read-ahead) are handled by the block I/O daemons (`biod`) on the client.

## **1.3 The NFS Locking Service**

The NFS locking service allows you to create advisory locks on files and file regions on local and remotely mounted file systems.

To make use of the NFS locking service, a programmer needs to understand how to use the `fcntl` system call or the `lockf` subroutine. Programmers should refer to `fcntl(2)` and `lockf(3)` in the ULTRIX Reference Pages for programming information.

File locking is a way to manage shared file access. The basic steps a process takes when locking a file or region of a file are:

1. Seeing if the file or region within the file is locked
2. Applying a lock if the file or region is not locked
3. Making the changes to the file or region
4. Unlocking the file or region

The NFS locking service coordinates the dispersal of locks to local and remote file systems. The NFS locking service communicates with the

lock daemons on the network.

The NFS locking service uses a stateless approach to failure recovery. The fundamental element of this approach is that the status monitor detects both client and server failures and recoveries. This approach is passive. When the client status monitor detects that a failed client has reinitialized (recovered), it notifies the local locking daemon of the failure. The lock daemon then activates the appropriate recovery mechanism.

If the NFS server fails and the NFS locking service is enabled, all the locks managed by the server's local processes are lost. However, when the server recovers, the lock manager daemons on the client systems send reclaim requests for the NFS locks. The server lock manager reestablishes the previously acquired locks associated with the reclaim requests, provided the requests are received within the grace period built into the NFS locking service. During the grace period the server lock manager honors only reclaim requests. Once the grace period expires, reclaim requests are no longer valid, and the server lock manager accepts only new requests. At this time, the server lock manager cannot reestablish old locks. Instead, the client lock manager can create new locks through the use of the interface primitives in the `fcntl` system call or the `lockf` subroutine.

If a client fails while it is using the NFS locking service, then when the client recovers, the status monitor daemon notifies the appropriate servers. The server lock manager then releases the locks. The client applications can then issue new lock requests as part of their recovery procedure.





## Setting Up the Network File System 2

This chapter explains how to set up a Network File System (NFS) using the `nfssetup` command. The `nfssetup` command allows you to set up your system as an NFS server, client, or both. You can also use the `nfssetup` command to enable or disable the NFS locking service. See Chapter 1 for a description of NFS and see `nfssetup(8nfs)` in the ULTRIX Reference Pages for further information about the `nfssetup` command.

The topics discussed in this chapter are:

- Preparing to Set Up NFS
- Setting up NFS automatically
- Setting up NFS manually
- Mounting a remote file system or directory
- Modifying the NFS environment

### 2.1 Preparing to Set Up NFS

Before you can set up NFS on your system, your system must be established on a local area network. In addition, you must know the answers to the following questions:

- Will your system be acting as a server? That is, will your system be exporting file systems and directories?

If your system will be acting as a server, then you must have the following information:

- How much of an NFS work load do you expect your system will get?

For an average work load, four `nfstd` daemons and four `biod` daemons (block input/output) are sufficient. You can configure more or less, depending upon the work load you anticipate for your system.

- What are the full pathnames of the file systems and directories your system will be exporting? The following is an example of a pathname:

`/usr/staff/public`

- Which hosts (clients) on the network do you want to allow access to your system's exported file systems and directories?

If your system will be running Yellow Pages (YP) in addition to NFS, you can allow network groups access to the exported file systems and directories in addition to, or instead of, hosts on your network. Network groups are described in the Guide to the Yellow Pages Service.

### **Note**

For security reasons, you should not allow a remote superuser access to your system unless the remote hosts (and superusers) on your network are trusted.

- Will your system be acting as a client? That is, will your system be accessing exported file systems and directories from other hosts (servers) running NFS on your network?

If your system will be acting as a client, then you must have the following information:

- What are the names of the remote hosts whose file systems and directories you will be importing (remote mounting on your local system)?
- What are the full pathnames of the remote file systems and directories you will be mounting from the remote host (server)?
- What are the full pathnames of the local mount points on your system where you will be mounting the remote file systems and directories? A common convention is to have the `nfssetup` command create a directory with the same name as the remote system, for example:

`/hisvax`

To run NFS on your system, NFS must be configured into your kernel (`/vmunix`). The ULTRIX operating system is distributed with the necessary code to configure NFS in the kernel. The `nfssetup` command checks your kernel to verify that NFS is configured properly and informs you if it finds configuration problems.

## 2.2 Setting Up NFS Automatically

Run the `nfssetup` command to set up NFS automatically. With the system in multiuser mode, type:

```
# /etc/nfssetup
```

Once `nfssetup` is running, it prints this information, followed by prompts:

Checking the kernel...

The `nfssetup` command configures the network file system (NFS) environment for your system. All systems using NFS facilities must run the Remote Procedure Call (RPC) port mapper daemon. An entry for this daemon is placed in the `/etc/rc.local` file along with entries for the optional daemons you select.

You will be asked a series of questions about your system. Default answers are shown in square brackets ([ ]). To use a default answer, press the RETURN key.

The `nfssetup` command first asks you if you want to enable the NFS locking service. If NFS is not already running on your system, the `nfssetup` command prompts you for information to set up NFS.

If NFS is already running on your system, the `nfssetup` command asks you if you want to modify the rest of your current NFS configuration. If you type `n` (no), the `nfssetup` command terminates without modifying the NFS configuration other than to enable or disable the NFS locking service. If you type `y` (yes), the `nfssetup` command prompts you for information to modify your NFS configuration. Whether you set up NFS or modify your existing configuration, answer each of the `nfssetup` command prompts. The last prompt gives you the option to save the information, exit, continue, or restart `nfssetup`. When the `nfssetup` command has completed making the required modifications, it prints this informational message:

```
***** NFS SETUP COMPLETE *****
```

If you choose to quit the `nfssetup` command without having it make the changes, it prints this message:

```
Nfssetup terminated with no installations made.
```

### Note

If necessary, you can press CTRL/C to terminate the `nfssetup` command before it has completed. Or, you can wait until the last `nfssetup` prompt, which gives you the chance to start the command over, exit the command without having the NFS environment modified, or exit the command with the NFS environment being modified according to the answers you supplied to the prompts.

After `nfssetup` completes, reboot the system.

## 2.3 Setting Up NFS Manually

This section describes how to set up your system manually as an NFS server for exporting file systems and directories, and helps you to understand how NFS works.

For an interactive setup with default answers, follow the automatic NFS setup procedure (see Section 2.2).

Both the client and the server must be connected to an Internet network for NFS to be able to run and for remote mounts to work.

To set up NFS manually, follow these steps:

1. Edit the `/etc/exports` file
2. Edit the `/etc/rc.local` file
3. Edit the `/etc/fstab` file
4. Reboot the system

If you are setting up your system as an NFS client only, issue the `mount` command as described in Section 2.3.3. The system must be in multiuser mode before you can mount a remote file system.

### 2.3.1 Edit the `/etc/exports` File

Add one entry per mount-point pathname of each file system or directory you want to export to the `/etc/exports` file. An NFS server can export only its own local file systems and directories. See `exports(5nfs)` in the ULTRIX Reference Pages for further information. For example, to export the file system `/usr/src/mybin` to the world, with no special permissions, the exports file should have an entry like this:

```
/usr/src/mybin
```

To export the same file only to a client named `orange`, the file should have an entry like:

```
/usr/src/mybin    orange
```

To deny NFS access to a file system or directory, remove its entry from the `/etc/exports` file. To have the removal take affect immediately, type the following:

```
# showmount -e
```

### 2.3.2 Edit the /etc/rc.local File

NFS servers must run the portmap, mountd, and NFS daemons. To have these daemons run automatically each time the system is brought to multiuser mode, add entries for them to the /etc/rc.local file. If you are enabling or disabling the NFS locking service, then you also need an entry for the nfssetlock command.

#### Note

Be sure you add the NFS daemon entries after any YP entries in the rc.local file. If no YP entries exist, be sure the NFS daemon entries are after the following entry:

```
/etc/ifconfig lo0 localhost
```

**2.3.2.1 Make an Entry for the portmap Daemon** – Make sure that the remote procedure call (RPC) port mapper is running. Check the /etc/rc.local file for the following entry, which starts the port mapper:

```
if [ -f /etc/portmap ]; then
    /etc/portmap; echo -n ' portmap' > /dev/console
fi
```

If this entry does not exist, add it to the /etc/rc.local file. See portmap(8nfs) in the ULTRIX Reference Pages for further information about the port mapper.

**2.3.2.2 Make an Entry for the mountd Daemon** – Make sure that the mountd daemon is available for an rpc call. The mountd daemon must be present for a remote mount to succeed. Check the /etc/rc.local file for the following entry, which starts the mountd daemon:

```
if [ -f /etc/mountd -a -f /etc/portmap -a -s /etc/exports ]; then
    /etc/mountd; echo -n ' mountd' > /dev/console
fi
```

If this entry does not exist, add it, or a similar one, to the /etc/rc.local file. See mountd(8nfs) in the ULTRIX Reference Pages for further information.

**2.3.2.3 Make Entries for the NFS Daemons** - Make sure that the `nfstd` and `biod` server daemons are available to provide the NFS service. A typical number of `nfstd` and `biod` daemons for a large processor such as a VAX 8800 is four of each. MicroVAX processors may need only two of each. Check the `/etc/rc.local` file for entries like these:

```
if [ -f /etc/nfstd -a -f /etc/portmap ]; then
    /etc/nfstd 4 & echo -n ' nfstd'      >/dev/console
fi

if [ -f /etc/biod ]; then
    /etc/biod 4 & echo -n ' biod'      >/dev/console
fi
```

If these entries do not exist, add them, or similar ones, to the `/etc/rc.local` file. See `nfstd(8nfs)` in the ULTRIX Reference Pages for further information.

If you want your system to receive notification in the event of an NFS server being shut down, you must run the `rwalld` daemon. Place the following entry in the `/etc/rc.local` file:

```
if [ -f /usr/etc/rwalld -a -f /etc/portmap ]; then
    /usr/etc/rwalld & echo -n ' rwalld'  >/dev/console
fi
```

After you have added a file system or directory name to the `/etc/exports` file, the file system or directory is ready for client systems.

**2.3.2.4 Make an Entry for the NFS Locking Service** - By default, the NFS locking service is disabled and only local locking is enabled. The best way to enable the NFS locking service is to use the `nfsssetup` command as described in Section 2.2. To enable the NFS locking service manually, be sure the following entries are in the `/etc/rc.local` file:

```
# %NFSLOCKSTART%
echo 'Enabling NFS Locking'      >/dev/console
[ -f /usr/etc/nfssetlock ] && {
    /usr/etc/nfssetlock on & echo 'nfs locking enabled' >/dev/console
}
[ -f /usr/etc/statd ] && {
    /usr/etc/statd & echo -n 'statd '      >/dev/console
}
[ -f /usr/etc/lockd ] && {
    /usr/etc/lockd & echo 'lockd'         >/dev/console
}
# %NFSLOCKEND%
```

If these entries do not exist, add them, or similar ones, to the end of the NFS entries in the `/etc/rc.local` file. They belong immediately before the following entry:

```
# %NFSEND%
```

The `nfssetlock` entry containing the `on` option turns on the NFS daemon-based locking service. See `nfssetlock(8nfs)`, `statd(8c)`, and `lockd(8c)` in the ULTRIX Reference Pages for further information.

### 2.3.3 Edit the `/etc/fstab` File

If you want to have certain remote directories or file systems mounted automatically each time your system goes to multiuser mode, place the appropriate entry in the `/etc/fstab` file. For example, if you want the file system `/usr/users` to be mounted automatically from an NFS server named `spice` onto the local mount point `/mnt`, place an entry similar to the following in the `/etc/fstab` file:

```
/usr/users@spice:/mnt:ro:0:0:nfs:bg,soft:
```

#### Note

Be sure to include the `bg` option in the `/etc/fstab` entry. The `bg` option causes remote mount requests to be tried once in the foreground and then retried in the background if the initial mount fails. Without the `bg` option, remote mount requests are made in the foreground, which prevents your system from rebooting if any server listed in `/etc/fstab` is not currently available.

See `fstab(5)` in the ULTRIX Reference Pages for information on the file format.

To mount a file system or directory immediately, without bringing the system to single-user and then multiuser mode, you can execute the `mount` command. The following example is a sample mount request made from an NFS client running in multiuser mode:

```
# mount -t nfs spice:/usr/src /spice/usr/src
```

See `mount(8nfs)` in the ULTRIX Reference Pages for further information about mounting remote file systems.

### 2.3.4 Reboot the System

After you have edited the `/etc/rc.local` and `/etc/fstab` files, you need to reboot the system. This causes the changes to the `rc.local` file to take effect. The following command line shuts down and reboots the system immediately:

```
# /etc/shutdown -r now
```

See shutdown(8) in the ULTRIX Reference Pages for information on how to reboot your system.

### Note

If your system is in multiuser mode and you want to execute the NFS-related, but not NFS locking-related, commands and daemons without having to bring the system to single-user and then multiuser mode, type the following commands:

```
# /etc/portmap
# /etc/mountd
# /etc/nfsd 4 &
# /usr/etc/rwalld &
# /etc/biod 4 &
```

### Note

Do not issue the nfssetlock, statd, or lockd commands while the system is in multiuser mode. To do so could cause the locking information to become lost during the transition from local (kernel-based) to NFS (daemon-based) locking.

## 2.4 Modifying the NFS Environment

After NFS is set up and running, you may choose to modify the NFS environment. For example, if your system is an NFS server, you may decide to export an additional directory or remove a client from your list. There are two ways to modify the NFS environment: manually, by editing the appropriate NFS files, or automatically, using the nfssetup command.

### 2.4.1 Modifying the NFS Environment Automatically

If you have only a few modifications, it is easiest to edit the files manually. For example, if your system is a server and you want to modify the systems to which it exports a file system, edit the /etc/exports file. This is described in Section 2.3.1. If your system is a client and you want to mount a remote file system automatically each time the system boots, edit the /etc/fstab file. This is described in Section 2.3.3.



### **2.4.2 Modifying the NFS Environment Manually**

If you have numerous modifications to make, it may be easier to use the `nfssetup` command. This command appends the new information you provide to the end of the appropriate files. However, it does not delete information that existed before you ran the `nfssetup` command.

To enable or disable the NFS locking service, it is best to use the `nfssetup` command as described in Section 2.2. However, you can enable or disable it manually by editing the `/etc/rc.local` file as described in Section 2.3.2.



## Managing the Network File System 3

This chapter explains how to manage the Network File System (NFS). The topics discussed in this chapter are:

- Mounting a remote file system or directory
- Programming with the NFS locking service
- Locking remote files
- Recovering NFS locks
- Addressing system security
- Identifying system differences

### 3.1 Mounting a Remote File System or Directory

Any system can act as a client and mount a remote file system or directory onto a local mount point, provided it can reach the NFS server over the network and its host or netgroup name is included in the server's `/etc/exports` file. The following shows two formats of the mount command to mount a remote file system or directory:

```
mount -t nfs server_name:/filesystem /mount_point
```

```
mount -t nfs filesystem@server_name /mount_point
```

Note that you can mount either an entire remote file system or any subdirectory within a remote file system that has an entry in the remote system's `/etc/exports` file. For example, to mount the reference pages from the remote host `pluto` onto the local directory `/mnt`, type:

```
# mount -t nfs pluto:/usr/ref /mnt
```

While a file system or directory is remotely mounted, it is treated as a file

system by the local system.

### Note

The mount-point directory must exist on the local system before you issue the mount command to mount a remote file system or directory. A common practice is to create a directory with the same name as the remote host. This makes it easy to keep track of where the remotely mounted file systems and directories reside.

You can mount a remote file system or directory with either the `soft` or `hard` option. If you use the `soft` option, NFS operations in that file system fail with an error code (`ETIMEDOUT`) after a prescribed number of tries if the server is down or slow to respond. If you use the `hard` option, NFS operations in that file system do not fail: they continue to try until they either succeed or are stopped. By default, remote file systems and directories are mounted with the `hard` option. See `mount(8nfs)` in the *ULTRIX Reference Pages* for further information.

To make sure you have mounted a file system or directory and that it is mounted where you expect it to be, use the `mount` command without an argument. This command displays the mounted file systems and directories:

```
# /etc/mount
/dev/ra0a on /
/dev/ra0g on /usr
spice:/usr on /spice type nfs (rw,soft)
```

To have frequently used file systems and directories mounted automatically at startup time, place an entry for them in the client system's `/etc/fstab` file. For example, the following entry causes the file system `/usr/users` on the remote host `spice` to be mounted automatically at startup time on the local system on `/tmp`:

```
/usr/users@spice:/tmp:ro:0:0:nfs:bg,soft:
```

### Note

Be sure to include the `bg` option in the `/etc/fstab` entry. The `bg` option causes remote mount requests to be tried once in the foreground and then retried in the background if the initial mount fails. Without the `bg` option, remote mount requests are made in the foreground, which prevents your system from rebooting if any server listed in `/etc/fstab` is not currently available.

format.

The following example is a sample mount request made from an NFS client:

```
# mount -t nfs sugar:/usr/src /sugar/usr/src
```

In this example, the client asks the server `sugar` to return a file handle (fhandle) for the directory `/usr/src`. An fhandle is a block of information that NFS uses to identify files and directories. The mount system call passes the fhandle to the client kernel. The kernel looks up the directory `/sugar/usr/src` and, if it exists and has the correct permissions and so forth, ties the fhandle to the directory in a mount record. From then on, all file system requests to that directory and below, go through the fhandle to the server `sugar`.

### 3.2 Programming With the NFS Locking Service

The NFS locking service allows you to write programs that place advisory read and write locks on local and remote files or file regions. Specifically, in the code you lock an offset and a certain number of bytes.

With the NFS locking service, programmers can apply the `lockf` and `fcntl` primitives to remote files.

With the NFS locking service, previously written programs that ran in a local environment using `fcntl` and `lockf` primitives will work in the local and remote environments, provided the NFS locking service is enabled. If primitives other than `fcntl` and `lockf` were used, the programs might not be as efficient in the remote environment as they could be if these primitives were used. In that case, you may want to rework the code to take advantage of the NFS locking service.

Without the NFS locking service you can only lock files and regions on the local system. However, with the NFS locking service you can lock both local and remote files and regions by using the `fcntl` system call and the `lockf` subroutine in your programs. These primitives synchronize file and region access among processes using the locking mechanism. Programmers should understand the `fcntl` and `lockf` primitives in order to write programs that take advantage of the NFS locking service. See `fcntl(2)` and `lockf(3)` in the ULTRIX Reference Pages for programming information.

There are two types of locks: read and write. Any number of processes can apply an additional read lock on a file or region that already has a read lock. A write lock is exclusive; that is, once a write lock is made, no other process can obtain read or write locks for that file or region until the server lock manager releases the write lock. A read lock prevents another process from obtaining a write lock.

You can request read and write locks to be either wait or no wait. A wait lock waits until it can obtain a lock before the process can continue. A no wait lock responds immediately, returning a status of lock granted or denied.

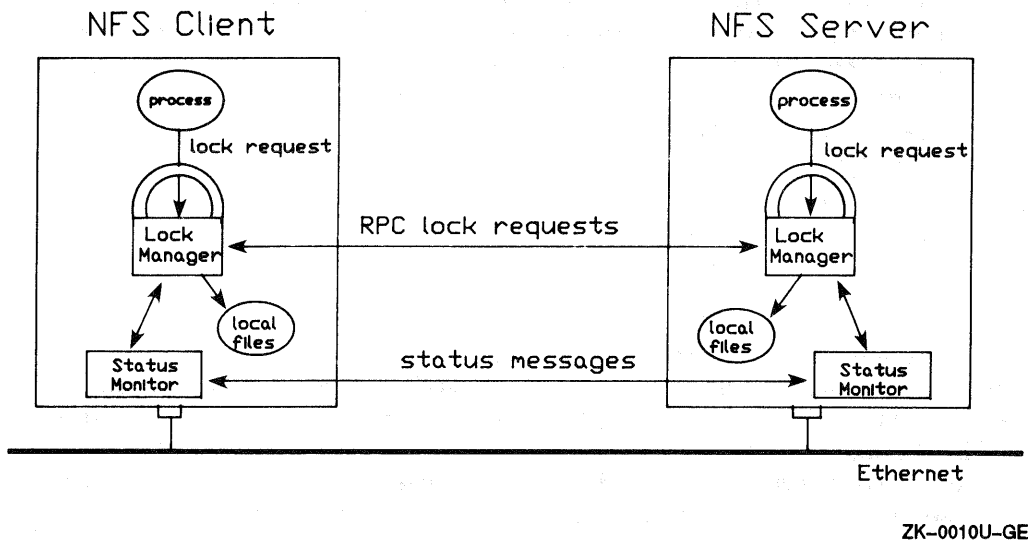
### **3.3 Locking Remote Files**

The NFS locking service allows processes to lock local and remote files or regions of files. The key elements for locking a remote file are:

- The client process requesting the lock
- The client lock manager
- The server lock manager
- The client status monitor
- The server status monitor

Figure 3-1 shows the relationship of each of the locking elements and illustrates the following steps for locking a file:

1. The client process requests locks from the local lock manager for both local and remote files or regions.
2. The client lock manager forwards the lock requests for remote data to the server lock manager, using the Remote Procedure Call (RPC) and the eXternal Data Representation (XDR) package.
3. The client and server lock managers request the status monitor service to monitor the sites participating in locking the remote file. The client lock manager requests to be notified of any change in status (up or down) of the server system. The server lock manager requests to be notified only if the client system comes up.



**Figure 3-1: Relationship of Client and Server with the NFS Locking Service**

### 3.4 Recovering NFS locks

The recovery procedure for NFS locks is passive. Rather than continually sending requests to see if clients and servers are up, the NFS locking service relies on the status monitor. The status monitor does not detect failures (system crashes). Instead, the status monitor detects system recoveries, that is, the status monitor detects whenever a system is brought on line or booted.

If a client system has locks on a remote file when the server goes down, the process accessing the remote file is interrupted. The server's lock manager loses its information of what locks were held by whom. The following protocol takes place to reconstruct the server's state information when the server recovers:

1. When the status monitor informs the client lock manager of the server's recovery, the client lock manager automatically sends a reclaim request to the server lock manager for each of the processes holding locks at the time of the failure. The reclaim request is a repeat of the original request that the process made before the server failed.
2. Upon recovery the server waits for a grace period. Typically, the grace period is about 45 seconds, but you can alter this by using the

-g option of the lockd subroutine. During the grace period, it grants any reclaim requests immediately. The reclaim requests arrive only from those client processes still interested in the locks. If a client process no longer needs a lock, it can cancel its original request. The client lock manager does not send reclaim requests to the server lock manager for cancelled locks.

3. During the grace period, the server lock manager reconstructs its state information from any reclaim requests received.
4. After the grace period, the server lock manager accepts new lock requests.

Server systems are unaware of a client's failure until the client recovers and informs the status monitor. The server's operations are not impacted by a client's failure. This is a feature of a stateless system. The server holds the client process' locks until the client reboots, or until the server reboots. Each time a server reboots, it releases all previously held locks. The server then goes through the recovery steps listed above.

See lockd(8c) in the ULTRIX Reference Pages for further information.

## **3.5 Addressing System Security**

NFS operates under the general assumption that you have a friendly network user community. Because this assumption may not always be true, this section explains how you can modify the security aspects of your system. The topics discussed in this section are:

- Superuser access over the network
- Mailing to superuser (root) across NFS systems
- Port monitoring
- Increasing system security

### **3.5.1 Superuser Access over the Network**

Under NFS, a server exports file systems and directories so that clients can mount them onto their local systems. Servers can only export file systems and directories that they own, and superuser privileges are not automatically extended to the superusers on client systems. By default, the superusers on client systems are given the category of "other" on the server. The superusers on clients are denied permission on remotely mounted file systems unless those file systems have read or write permissions set for the user group other.

Also, client superusers cannot change the ownership of remotely mounted files. Nonprivileged users cannot run the chown command and root is



treated as a nonprivileged user on remote access. Therefore, no one but root on the server can change the ownership of remotely mounted files, unless the server explicitly allows superuser access to remote client systems.

### **Note**

For security reasons, you should not allow a remote superuser access to your system unless the remote hosts and superusers on your network are trusted.

In a friendly network environment, you can allow superuser access over the network. To allow this access, use the `-r` option in the `/etc/exports` file. The `-r` option maps the remote superuser's identification to the number assigned. For example, the following entry in the `/etc/exports` file allows the superuser of the client system `spice` to access to the server's `/usr/games` file system:

```
/usr/games -r=0  spice
```

In this example, the superuser's identification (uid) is 0. Because the superuser (root) uid is always 0, you will usually want to use 0 with the `-r` option. See `exports(5nfs)` in the ULTRIX Reference Pages for further information.

### **3.5.2 Mailing to Superuser (root) Across NFS Systems**

The root restriction described in Section 3.5.1 prevents clients from mailing to superuser (root) on the server while `/usr/spool/mail` is remotely mounted from the server. For example, if a client system, `spice`, is using `/usr/spool/mail`, which is remotely mounted from the server, `sugar`, mailing to root on `spice` may not work as expected because of the restriction on root access across NFS.

The result of mailing to root in this way causes one of the following two conditions:

- If there is no root mailbox (`/usr/spool/mail/root`) on the server system, `sugar`, the file is created with an ownership of `-2`. This allows all root users that import the file system to read the root mailbox on the server. That is, root from the client system, `spice`, can read and delete mail for root on the server `sugar`.
- If a `/usr/spool/mail/root` mailbox exists on the server, sending mail to root on the client causes a notification of the server's root mail on the client. However, root on the client, which imports `/usr/spool/mail` from the server, cannot read the mailbox file.

You can work around these two problems by using the following convention instead of mailing to root:

On each system, set the aliases root and admin to the login name or names of the system administrators for that system. All mail intended for the administrators of that system should then be addressed to:

`admin@system`

This allows the administrators to send and receive mail between the server, sugar, and the client, spice, without confusion as to the access rights of the root mailbox. In effect, a `/usr/spool/mail/root` mailbox is not created or used.

To implement the convention, follow these steps:

1. Add the alias name admin (if not already present) to the line in `/usr/lib/sendmail.cf` that looks like:

```
CN MAILER-DAEMON postmaster
```

The line becomes:

```
CN MAILER-DAEMON postmaster admin
```

This adds the name admin to the class N.

2. Refreeze the sendmail configuration file:

```
# /usr/lib/sendmail -bz
```

3. Kill and then restart the sendmail daemon process while the system has little activity. If possible, it is best to do this in single-user mode. Use the `ps` command to obtain the process ID for sendmail. In the following example, the process ID is 2589:

```
# ps -ax | grep sendmail
# kill -9 2589
# /usr/lib/sendmail -bz
# /usr/lib/sendmail -bd -qlh -om
```

See `sendmail(8)` in the ULTRIX Reference Pages for further information about the sendmail command.

4. In the `/usr/lib/aliases`, file, add the login names of the system administrators and redefine (alias) the name root to be admin. For example, if the login names for the system administrators are john, mary, and joe, here are acceptable entries:

```
# aliases for users in the format:
#
#   alias:login
#   or
#   alias:system!login
#   or
#   alias:login@system
#
admin:john,mary,joe
root:admin
```

The entries admin:john,mary,joe and root:admin ensure that mail addressed to admin or root on the client system (spice) is redirected to the users john, mary, and joe. The problems associated with mailing to the login name root are avoided.

5. Issue the newaliases command to bring the logical destinations for root-bound mail into effect:

```
# newaliases
```

All systems in the local area network should follow this convention. As described here, mail for root or admin on any system can be automatically directed to any user login on any system.

### 3.5.3 Port Monitoring

Only privileged users can attach to some Internet domain source ports. These are known as privileged ports. NFS does not check to see if a client is bound to one of these. You may want to activate NFS server port checking to ensure that file access requests were generated by the client kernel rather than forged by an application program. To activate NFS server port checking, type:

```
# /etc/nfsportmon on
```

To turn source port checking off, type the nfsportmon command again, substituting off for on.

### Note

Although the ULTRIX operating system enforces the privileged port convention, some operating systems do not. If hosts running one of these operating systems are on your network, activating port checking might not improve security, but could prevent those systems from functioning properly as NFS client systems.

### 3.5.4 Increasing System Security

Any time a server allows its file systems and directories to be exported, it opens a degree of insecurity. The only way to maximize system security is not to export any file systems or directories. To do this, delete the `/etc/exports` file and make sure that no NFS daemons ( `nfsd` ) are running. Your system will not be an NFS server, but it can still be a client.

If you decide to have your system export file systems and directories, you can increase system security by limiting the client systems that are allowed to mount them remotely.

The `/etc/exports` file defines an export list for each of the file systems and directories that a client can mount. When a client makes a mount request to an NFS server, the server checks to see if the client's name appears in its export list. If no remote system (client) names are specified for a given exportable file system or directory, then any client on the network can mount that file system or directory. If one or more client names are listed for a file system or directory, then only those clients specified can mount remotely the server's exported file system or directory.

The following example from Chapter 1 shows sample `/etc/exports` file entries:

```
/usr/staff/doe green      # export directory only to host green
/usr/staff -o pink        # export file system read only to host pink
/usr/local                # export file system to the world
/usr2 pink green black    # export file system only to these hosts
```

Note that there should be only one entry per file system or directory being exported. Multiple entries are not supported. In this example, the file system `/usr/local` can be mounted remotely by any of the NFS client systems on the network. The file system `/usr/staff`, however, can be mounted remotely only from the client `pink`. The directory `/usr/staff/doe` can be mounted remotely only from `green` and `pink`.

An entry in the `/etc/exports` file for a given directory exports that directory and all subdirectories in it, except for those subdirectories that reside in a different file system (disk partition) than the exported directory.

You can use the `fsrand` command to increase the security of a file system exported by NFS. This command creates random inode generation numbers on all the inodes of a file system, which makes it much more difficult for a client to forge the pointer to a file system or directory. However, before using the `fsrand` command, you must unmount the file system and then run the `fsck` command on it. For example, to use `fsrand` on the file system `/dev/ra0g`, here is the command sequence:

```
# /etc/umount /dev/ra0g
# fsck /dev/rra0g
# fsirand /dev/rra0g
```

The `fsirand` command invalidates any mounts, remote or local, to the file system being modified. Therefore, you may want to warn NFS client systems before you issue the `fsirand` command. The following command tells you what client systems have your file systems or directories remotely mounted:

```
# /usr/etc/showmount -a
```

Before the `fsirand` command is issued, any NFS clients should unmount the file system being affected. After the `fsirand` command has completed, the client systems can then mount the file system again. If a client does not unmount and mount the file system, an error message similar to the following appears on that client system when a remote user tries to access the file system:

```
NFS server: stale fhandle fs(9,0) file 1803 gen 8
local gen 9 nlink 2, client address = 128.45.40.19
```

The solution is to unmount the file system on the client and then mount it again.

See `fsirand(8nfs)` in the ULTRIX Reference Pages for further information.

NFS servers use the standard ULTRIX file access protection scheme. This protection scheme can protect files from all users except root. An NFS client sends user and group IDs along with an NFS file access request. The server uses this information to allow or disallow the request. Because a client superuser can assume the identity of any other user on the client system by substituting the user ID, that client superuser could have the access rights of another user on the server. Thus, the only way to protect sensitive exported data on the server is to make the data files owned by root and then disallow superuser access over NFS.

Exporting specific directories to specific client hosts provides more security than does exporting an entire file system to all the client hosts. For example, if a file system contains login directories for a number of users, each login directory can be exported to a specific list of hosts or to none at all. A client host with access to one of these directories does not have access to another exported directory, unless it was in the other directory's export list or the export list for the entire file system.

### 3.6 Identifying System Differences

Some processes work differently on remote file systems than they do on a local file system, and some processes do not work at all. This section

problems that might arise.

### 3.6.1 Some File Operations Fail

The flock primitive cannot lock a remote file. However, the lockf and fcntl primitives can lock remote files, provided the NFS locking service is enabled.

Append mode and atomic writes are not always reliable on remote files that are being written to by more than one process at a time. In the worst case, data written in append mode can be lost. Data written in large pieces, generally greater than 8K bytes, can be interleaved with data written by other processes. You should not use NFS to operate on files that are likely to be modified by several users simultaneously.

### 3.6.2 System (Clock) Times Can Vary

Because each operating system on the network keeps its own time, the clocks between the NFS client and server can become unsynchronized. This could introduce a problem in some situations if the clocks vary by more than an hour (60 minutes). The problems arise because modify, create, and access time attributes of a remote file are determined by the server's clock rather than the client's clock. See stat(2) in the ULTRIX Reference Pages for further information. The following examples show two potential problems and how the ULTRIX operating system compensates for them:

- A few programs, such as ls, make the assumption that an existing file could not have been created in the future. Here are the two basic forms of ls output, based upon how old the files being listed are:

```
client# date
Jan 22 15:27:01 PST 1988
client# touch test2
client# ls -l file*
-rw-r--r--  1 root          0 Dec 27  1987 test1
-rw-r--r--  1 root          0 Jan 22 15:27 test2
```

In this example, the first output line from the ls command shows the month, day, and year of the last modification of the file test1, because test1 is more than six months old. The second output line from the ls command shows the month, day, and time of the last modification of the file test2 because test2 is less than six months old.

The ls command calculates the age of a file by subtracting the modification time of the file from the current time. If the result is

Assume that the time on the server is Jan 22 16:28:01 (61 minutes ahead of the client system's time):

```
server# date
Jan 22 16:28:01 PST 1988
server# touch test3
server# ls -l file*
-rw-r--r--  1 root          0 Dec 27  1987 test1
-rw-r--r--  1 root          0 Jan 22  15:27 test2
-rw-r--r--  1 root          0 Jan 22  16:28 test3
```

The following example is from the client system:

```
client# date
Jan 22 15:27:01 PST 1988
client# ls -l file*
-rw-r--r--  1 root          0 Dec 27  1987 test1
-rw-r--r--  1 root          0 Jan 22  15:27 test2
-rw-r--r--  1 root          0 Jan 22  1988 test3
```

Note that this example shows test3 as having a creation date of Jan 22 1988. This is because as far as the client is concerned, test3 was created more than 60 minutes in the future.

In the following example, note that the file created on the client system (test4) acquires the creation date of the server system:

```
client# touch test4
client# ls -l file*
-rw-r--r--  1 root          0 Dec 27  1987 test1
-rw-r--r--  1 root          0 Jan 22  15:27 test2
-rw-r--r--  1 root          0 Jan 22  1988 test3
-rw-r--r--  1 root          0 Jan 22  1988 test4
```

The problem of clock skew results because the difference between the two times is huge:

$$\begin{aligned} &(\text{current time}) - (\text{modification time}) = \\ &(\text{current time}) - (\text{current time} + 61 \text{ minutes}) = -61 \text{ minutes} \end{aligned}$$

In binary representation, -61 minutes is greater than six months, causing the ls command to believe the new file was created long ago.

In most cases, server and client systems have less than 60 minutes of clock skew. The ls command recognizes this and prints the times correctly if the time appears to be 60 minutes or less in the future.

- The ranlib command was also modified to deal with clock skew. The ranlib command timestamps a library when it is produced, and the ld command compares the time stamp with the last modified date of the library. If the last modified date occurred after the time stamp, then ld instructs the user to run ranlib again, and reload the library.

If the library lives on a server whose clock is ahead of the client that runs `ranlib`, `ld` always gives warning messages. The `ranlib` command now sets the time stamp to the maximum value of the current time and the library's modify time.

Remember, if your application depends upon both local time and the create, modify, or access time attributes of files, then it may encounter clock skew problems when used with an NFS file system or directory.



## Troubleshooting the Network File System 4

This chapter describes the most common causes of NFS malfunctions and provides some methods for solving the problems. It also lists the NFS error messages and provides remedies for each.

The source of an NFS problem usually lies in one of the following areas:

- The network access control policies do not allow the operation
- The requested function does not exist
- The client software or environment is down
- The server software or environment is down
- The network is down

However, before you can solve NFS problems, you must be familiar with how NFS operates and you should be familiar with the following NFS commands, daemons, and files: `mount`, `nfsd`, `biod`, `showmount`, `rpcinfo`, `mountd`, `nfsstat`, `fstab`, and `exports`. For additional information, see Chapter 2.

When solving NFS problems, keep in mind that there are three main points of failure: the server, the client, or the network itself. The following troubleshooting strategy shows how to isolate each individual component to find the one that is not working properly.

### Note

The client and the server must be connected by a network for NFS to be able to run and for remote mounts to work.

### 4.1 Network or Server Failures

In the event of network or server problems, programs that access hard-mounted remote files fail differently from those that access soft-mounted remote files. Hard-mounted remote file systems cause programs to retry indefinitely, until the server responds. Soft-mounted remote file systems return an error after trying either the default number of times or the number of times specified by the `retrans` argument in the `mount` command. If neither `hard` nor `soft` is specified in the `mount` command, `hard` is the

default. See `mount(8nfs)` in the ULTRIX Reference Pages for further information.

Once the `mount` command with the `hard` option has succeeded, programs that access the hard-mounted files block as long as the server fails to respond. In this case, NFS prints a message to the console and to the error logger in the following format:

NFS server *hostname* not responding, still trying

Once the `mount` command with the `soft` option has succeeded, if the server fails to respond to a request, NFS prints a message to the console and to the error logger in the following format:

NFS operation failed for server *hostname*, Timed out

If a client is having NFS problems, check first to make sure that the server is up and running. For example, if the server's name is *sugar*, type the following from the client:

```
# /etc/rpcinfo -p sugar
```

If the server is up, `rpcinfo` lists the program, version, protocol, and port numbers in this way:

program	vers	proto	port	
100004	2	udp	1027	ypserv
100004	2	tcp	1024	ypserv
100004	1	udp	1027	ypserv
100004	1	tcp	1024	ypserv
100007	2	tcp	1025	ypbind
100007	2	udp	1035	ypbind
100007	1	tcp	1025	ypbind
100007	1	udp	1035	ypbind
100003	2	udp	2049	nfs
100005	1	udp	1091	mountd

If `rpcinfo` is able to produce this list, you can use `rpcinfo` to check if the `mountd` server is running. For example, if the previous list is the output from the server *sugar*, type:

```
# /etc/rpcinfo -u sugar 100005 1
```

If the `mountd` server is running, `rpcinfo` should respond with:

```
program 100005 version 1 ready and waiting
```

If these two `rpcinfo` commands fail, try the following:

- Log in to the server and see if it is running properly. If the NFS server daemons `/etc/portmap`, `/etc/mountd`, and `/etc/nfsd` are not running, you may need to set up NFS. For information about setting up NFS, see Chapter 2.

- If the server is running properly, but your system cannot reach it, check the Internet connections between your system and the server.
- If you cannot remotely log in to the server using rlogin, but the server is up, check your Ethernet connection by trying to log in remotely to another host on the network. You should also check the server's Ethernet connection.

### Note

You do not need biod or any NFS server daemons running to be an NFS client.

The following sections describe the most common types of failure. The first two sections address failures during a remote mount. The remaining sections describe potential problems once file systems and directories are mounted.

## 4.2 Remote Mount Failures

If the mount command fails to mount the remote file system or directory, check this section for details about how remote mounts operate. This information may help you discover how your remote mount failed and what you can do to remedy the situation.

The mount command gets its parameters from either the command line or the file /etc/fstab. See Chapter 3 for further discussion of how to mount a remote file system or directory. The following is a sample mount command:

```
# mount sugar:/usr/src /sugar/usr/src
```

Here are the steps the mount command takes to mount a remote file system on sugar. The remote mount can fail during any one of these steps:

1. The mount command checks the mount table to be sure that the requested mount has not already been done.
2. The mount command parses the first argument into host sugar and the remote directory /usr/src.
3. If YP is running, the mount command calls the YP binder daemon, ypbind, to determine which server system has the YP server. The mount command then calls the ypserv daemon on that system to get the Internet protocol (IP) address of sugar. If YP is not running, the IP address is obtained from the local /etc/hosts file.
4. The mount command calls sugar's port mapper to get the port number of sugar's mountd daemon.

5. The mount command calls sugar's mountd daemon and passes /usr/src to it.
6. Host sugar's mountd daemon reads the /etc/exports file and looks for an entry that allows access for /usr/src.
7. If YP is running, sugar's mountd daemon calls the YP server, ypserv, to expand the host names and netgroups (discussed in the Guide to the Yellow Pages Service) in the export list for /usr/src. If YP is not running, the mountd daemon uses the /etc/hosts file to expand the export list. The expanded export list is checked to see if the client has access authorization for /usr/src.
8. Host sugar's mountd daemon performs an nfs\_getfh system call on /usr/src to get the fhandle, which was previously referred to as a pointer to a file system.
9. Host sugar's mountd daemon returns the fhandle.
10. The mount command checks to see if /sugar/usr/src is a directory.
11. The mount command performs a mount system call with the fhandle and /sugar/usr/src.
12. The mount system call adds an entry to the mount table.

### 4.3 Remote Mount Error Messages

The error messages in this section are arranged according to where the types of failures can occur during the mount procedure and they show the error message labels you are most likely to see.

nfs\_mount: cannot mount xxx on yyy: Mount device busy

The file system or directory you are trying to mount is already mounted.

nfs\_mount: illegal file system name "xxx";  
use host:pathname

You may have forgotten to specify the name of the server when you issued the mount command. For example, to mount the file system /usr/src from the server sugar, type:

```
server# mount sugar:/usr/src /sugar/usr/src
```

Don't know how to mount xxx

If mount is called with a directory or file system name but not both, it looks in the /etc/fstab file for an entry whose file system or directory field matches the argument. If you see this error message, there is no entry for the argument you specified on the mount

command line in the `/etc/fstab` file.

For example, the following command causes `mount` to search `/etc/fstab` for an entry that has a mount-point name of `/sugar/usr/src`:

```
# mount /sugar/usr/src
```

The `mount` command looks for this type of entry in the `/etc/fstab` file:

```
/usr/src@sugar:/sugar/usr/src:rw:0:0:nfs:soft,bg:
```

If no entry exists in the `/etc/fstab` file, then the `mount` command cannot perform the operation. However, if the `mount` command finds a suitable entry, it performs the mount as if you had typed the following:

```
# mount -o soft,bg sugar:/usr/src /sugar/usr/src
```

`/etc/fstab`: No such file or directory

The `/etc/fstab` file does not exist. The `mount` command discovered this when it tried to look up the name specified on the command line.

`xxx` not in hosts database

There is no entry in the `/etc/hosts` file for the NFS server specified in the `mount` command line. If YP is running, then there is no entry in the hosts YP map for the host name specified.

`nfs_mount`: invalid directory name "`xx`"  
directory pathname must begin with `/'`.

The mount point on the local (client) system must be an absolute path starting at the root directory (`/`).

`nfs_mount`: `xxx` server not responding: port mapper failure - rpc timed out  
Giving up on `yyy`

The server you are trying to mount from is down, or its port mapper is inoperative. Try to log in remotely to the server. This proves the network is up. If you are able to log in, execute the `rpcinfo` command from the server. The following example uses `sugar` as the name of the server:

```
server# /etc/rpcinfo -p sugar
```

If the port mapper is running properly on the server, `rpcinfo` lists the registered program numbers. If it does not, restart the port mapper on the server. You also need a port mapper running on the client host. If it is not running there, start it.

After you have restarted the port mapper, kill and restart the `mountd` and `rpcd` daemons on the client. If YP is running, kill and restart

the ypbind daemon on the server as well. To do this, find the process IDs of the portmap and mountd daemons, and of the ypbind daemon if YP is running:

```
server# ps ax | grep portmap
 69 ? I    0:28 /etc/portmap
632 p1 S    0:00 grep portmap

server# ps ax | grep mountd
 86 ? I    0:28 /etc/mountd
634 p1 S    0:00 grep mountd

server# ps ax | grep ypbind
 80 ? I    0:28 /etc/ypbind
636 p1 S    0:00 grep ypbind
```

Next, kill the daemons on the server using the kill command and specifying the process IDs. Note that in the following example the ypbind daemon is also killed because YP is running:

```
server# kill -9 69 86 80
```

Then, restart the daemons on the server:

```
server# /etc/portmap
server# /etc/mountd
server# /etc/ypbind
```

You can also restart the daemons by first bringing the server to single-user mode and then to multiuser mode, if you prefer.

nfs\_mount: xxx server not responding: rpc prog not registered

The mount got through to the port mapper, but the NFS mountd daemon was not registered. Log in to the server and check to be sure that the file /etc/mountd exists. Then, run the ps command to see if the mountd daemon is running. If it is not running, restart it by typing:

```
server# /etc/mountd
```

nfs\_mount: cannot mount xxx on yyy: No such file or directory

The local directory does not exist. Check the spelling. Try to list the files in both directories, using the ls command.

nfs\_mount: access denied for yyy

Your host name is not in the export list for the file system or directory you want to mount from the server. If your server's host name is sugar, you can get a list of its exported file systems and directories by typing:

```
# /usr/etc/showmount -e sugar
```

If the file system or directory you want to mount remotely is not in the list, or if your host or netgroup name is not in the user list for the file system or directory, log in to the server and check the `/etc/exports` file for the correct file system entry. A file system or directory name that appears in the `/etc/exports` file, but not in the output from `showmount` indicates a failure in `mountd`. The `mountd` daemon could not parse that line in the file, could not find the file system or directory, or the file system or directory name was not a locally mounted file system. If the `/etc/exports` file seems correct and YP is running, check the server's `ypbind` daemon; it may be dead or hung.

See `exports(5nfs)` in the ULTRIX Reference Pages for further information.

`nfs_mount: cannot mount xxx on file: Not a directory`

Either the remote or local path is not a directory. Check the spelling and try to list both directories, using the `ls` command.

`nfs_mount: cannot mount xxx on yyy: Not owner`

You must mount the remote file system or directory as superuser (root) on your system.

## 4.4 Client Programs Block

Client programs can block while doing file-related work if one of the following conditions exist:

- The NFS server is down
- The  `nfsd`  daemon is malfunctioning
- Two or more processes are deadlocked

### 4.4.1 The NFS Server is Down

If your NFS server is down, you might see this message on the user's terminal on the client system and in the error logger:

NFS server *hostname* not responding, still trying

This message includes the host name of the NFS server that is down. In this situation, there is probably a problem with one of your NFS servers or with the Ethernet connection or wire. See `uerf(8)` in the ULTRIX Reference Pages for information about the error logger.

If a program blocks because one or more NFS servers is down, the program will continue automatically when the server or servers come back up, provided the remote file systems and directories are hard-mounted. In this case, a message in the following format is displayed on the user's terminal and in the error logger:

NFS server *hostname* ok

No file damage will be incurred.

Operations on soft-mounted remote files from a down server time out and fail with the ETIMEDOUT error code. This usually causes the program that attempted the operation to terminate. You can specify the length of time and the number of retries before the operation fails as arguments to the mount command. See mount(8nfs) and intro(2) in the ULTRIX Reference Pages for further information.

#### **4.4.2 The nfsd Daemon is Malfunctioning**

If all of the servers are running, ask other users of the server or servers that you are using if they are having trouble. If more than one system is having problems getting service, then it is probably a problem with the server's nfsd daemon. Log in to the server and use the ps command to see if nfsd is running and accumulating CPU time. If it is running but not accumulating CPU time, kill it and then restart the nfsd daemon. If this does not work, reboot the server.

If no one else is experiencing problems with the server, check your Ethernet connection and the connection to the server.

#### **4.4.3 Two Or More Processes Are Deadlocked**

If your program blocks for a long period of time and it does not appear to be a problem with NFS, the process could be deadlocked. Killing the process is the only way to remove a deadlock. Use the ps command to get the process number. Then kill the process with the kill command.

### **4.5 System Hangs Part Way Through Boot**

If your system comes up partially after a boot, but hangs where it would normally mount remote file systems and directories, probably at least one server is down and the background option (bg) is not specified in the fstab entry, or your network connection is bad. See Chapter 2 for information about placing entries in the /etc/fstab file.

If you are running YP, your system will hang part way through the boot if all of the YP servers are down and your system does not have access to key system files such as /etc/hosts. At least one YP server must be up



and running if any system files are served by YP. See the Guide to the Yellow Pages Service for information about YP.

### Note

If you are running YP or BIND in addition to NFS, be sure that the YP and or BIND entries precede the NFS entries in the `/etc/rc.local` file.

You can leave the system in the hung state, and when all the servers listed in the `/etc/fstab` file are up on the network, your system will continue booting. If you want to have your system continue booting despite the down servers, you can do the following:

1. Halt the system.
2. Boot the system to single-user mode and run the `fsck` command on the local file systems.
3. Edit the `/etc/fstab` file and add the background option (`bg`). A sample `fstab` entry with the background option might be:

```
/usr/src@sugar:/sugar/usr/src:rw:0:0:nfs:hard,bg:
```

4. Reboot the system:

```
# /etc/reboot
```

If the `bg` option is specified in the `fstab` file entry, the remote file system or directory is automatically mounted when the server comes up on the network and begins functioning as an NFS server.

## 4.6 Slow Remote File Access

If access to remote files seems unusually slow, check the NFS server. If the server seems normal and other people are getting good response, make sure that the block I/O daemons are running on the client. Run the `ps` command to find the process IDs of the `biod` daemons. For example:

```
# ps ax | grep biod
 81 ? I    0:28 /etc/biod
630 p1 S   0:00 grep biod
```

If no `biod` daemons are running, start some. The `biod` daemons are not necessary, but they do perform read-ahead and write-behind on the client side, which may make I/O faster. In the following example, four are started:

```
# /etc/biod 4 &
```

MicroVAX processors may need only two biod daemons running.

Next, check your Ethernet connection. The following command tells you if your system is dropping packets:

```
# /usr/ucb/netstat -i
```

You can use the `-c` option with the `nfsstat` command to determine how much retransmitting a client is doing. A retransmission rate of 0.5% is considered high and may indicate a timeout value that is too small. Excessive retransmission may also indicate a bad Ethernet board, a bad Ethernet tap, a mismatch between board and tap, or a mismatch between your Ethernet board and the server's board. A significant level of bad transmissions (for example, if `badxid` is greater than 0.1%) indicates that the timeout value selected in the mount operation is too small. Try changing the timeout parameter and see what effect this has on the `retrans`, `badxid`, and `timeout` parameters. Setting the file system timeout value higher may actually increase throughput. See `mount(8nfs)` in the ULTRIX Reference Pages for information about the timeout parameter.

#### Note

The symptoms of a bad Ethernet board include garbled I/O data. This causes NFS to drop packets because the UDP checksums are bad. This in turn, causes NFS operations to time out. Type the following command to see if any UDP packets have been dropped:

```
# netstat -s
```

See `netstat(1)` and `nfsstat(8nfs)` in the ULTRIX Reference Pages for further information.

## 4.7 NFS Console Error Messages

The following error messages may be displayed on the NFS client system console and in the error logger. They note an NFS file access failure.

NFS server *hostname* not responding, still trying

File operations in a hard-mounted file system have suspended because communication between the client and the server has stopped.

NFS server *hostname* ok

File operations have resumed.

NFS *file operation* failed for server *hostname*: *reason*

If the operation is in a soft-mounted file system, and the

down, the reason for the failure is that the operation timed out.  
NFS write error, server *hostname*, remote file system full

A write operation failed because the remote file system is full.  
NFS write error *errno*, server *hostname*, fs(*n,n*), file *file*

A write operation was refused by the server. The fs and file variables are parts of the fhandle. See *errno(2)* in the ULTRIX Reference Pages for a description of write errors.

The following error messages might occur on the NFS server console and in the error logger:

NFS server: unexported fs(*n,n*) file *file*, client address = *n.n.n.n*.

The file system or directory in question has no entry in the */etc/exports* file.

NFS server: fs(*n,n*) not mounted, client address = *n.n.n.n*.

The file system with the major and minor numbers given (fs(*n,n*)) is not mounted on the server. Make sure that the appropriate file system is mounted on the NFS server. If it is mounted on the same device, for example */dev/ra0g*), then the client system can retry the operation. If the file system is mounted on a different device than when the client originally mounted it, have the client system unmount and then remount the remote file system.

NFS server: unexpected fs(*n,n*) file *file*

The file system with the major and minor numbers specified is not exported to the client. To export that file system it must have an entry with the proper permissions in the */etc/exports* file on the server.

NFS server: stale file handle fs(*n,n*) file *file* gen *n*  
local gen *n* nlink *n*, client address = *n.n.n.n*

The client system sends the information on the first line of this message to the server. The second line is information the server already has. In this message, fs is the major and minor device numbers (respectively) of the disk partition on which the file system resides, file is the inode number and gen is the generation number of the file described by the stale file handle. The local gen variable is the server's file generation number, and nlink is the number of links the server has to the file. If nlink is zero, then the file (inode) is unallocated on disk. The address variable is the Internet address of the client from which the stale file handle originated.

This error message is generated for one of two reasons:

- The file generation number does not match the local generation number on the server.
- The file identified by the file variable is not allocated; that is, there are no links.

The following can cause the above two situations:

- A client is operating on a file that has been removed by some other process. This is the most likely explanation if a single message or a small number of messages from the same client appears.
- The file system was recreated, for example with the `mkfs` command.
- The `fsirand` command has been run on the remotely mounted file system.
- The remotely mounted file system has been moved to a different device, for example, an RA60 pack was removed and replaced with another.

The solution to the last two situations listed above is to have all of the clients unmount and then remount the remote file system.

NFS server: couldn't get `fs(n,n)` file *file*, client address = *n.n.n.n*

A system error occurred while trying to access the file within the file system with the major and minor numbers (`fs(n,n)`) given. This is either a hardware disk error or an internal system error. Contact your DIGITAL Field Service representative.

# Index

## A

**aliases file**  
entries, 3-8

## B

**biod daemon**  
getting process ID, 4-9e  
NFS client and, 4-3n  
starting, 2-6

## C

**client**  
*See also* server  
defined, 1-1  
server and, 1-2

**client host**  
*See* client

**client program**  
hanging  
blocking, 4-7

**clock**  
unsynchronized server and client,  
3-12 to 3-14

**configuration file (sendmail)**  
editing, 3-8  
refreezing, 3-8

## D

**daemon**  
biod, 2-1  
nfsd, 2-1, 3-10  
RPC, 2-3  
sendmail, 3-8

**directory**  
limiting to specific clients, 1-3

## E

**error message (NFS mount), 4-4 to 4-7**

**error message (NFS)**  
console, 4-10 to 4-12

**Ethernet board**  
symptoms of bad, 4-10

**export**  
defined, 1-1

**exports file**  
adding entries, 2-4  
entries, 1-3e, 3-10e  
system security and, 3-10

## F

**fcntl**  
NFS locking and, 1-5

## **fhandle**

- defined, 3-3
- getting, 4-4
- returning, 4-4

## **file system**

- See* remote file system
- compared to remote file system,  
3-11 to 3-14
- limiting to specific clients, 1-3

## **flock primitive**

- remote files and, 3-12

## **fsirand command**

- system security and, 3-10

## **fstab file**

- bg option and, 2-7n, 3-2n, 4-8
- entry, 1-3e, 3-2, 4-9e, 2-7

## **L**

### **lock**

- no wait, 3-3
- read, 3-3
- types of, 3-3
- wait, 3-3
- write, 3-3

### **lock recovery**

- client failure, 3-6
- server failure, 3-5

### **lockf**

- NFS locking and, 1-5

## **M**

### **mail**

- aliasing root, 3-8
- sending to remote superusers, 3-7 to  
3-9

### **mount command**

- fstab file and, 4-5

### **mount command (NFS)**

- failure, 4-3 to 4-4

### **mountd daemon**

- starting, 2-5

## **N**

### **Network File System**

- See* NFS

### **NFS**

- defined, 1-1
- introduction
- modifying environment, 2-8
- service, 1-3
- setting up automatically, 2-3
- setting up manually, 2-4 to 2-8
- setup preparation, 2-1
- setup requirements, 2-2
- troubleshooting, 4-1 to 4-12

### **NFS daemons**

- YP daemons and, 2-5n

### **NFS lock**

- recovering, 3-5

### **NFS locking**

- client and server, 3-4f
- client failure and, 1-5
- defined, 1-4
- programming with, 3-3
- recovery, 1-5
- server failure and, 1-5

### **NFS locking service**

- See* NFS locking

### **NFS Locking Service**

- description, 1-4 to 1-5

### **NFS server**

- recovering, 4-8

### **nfsd daemon**

- starting, 2-6

### **nfssetup command**

- defined, 2-1
- terminating, 2-3

**no wait lock**

*See lock*

**P**

**port**

*See also* privileged port  
monitoring, 3-9  
privileged, 3-9

**port mapper**

defined, 1-4  
starting, 2-5

**privileged port**

other operating systems and, 3-9n

**R**

**rc.local file**

NFS entries, 2-6  
with YP and NFS entries, 4-9n

**read lock**

*See lock*

**remote directory**

*See* remote file system

**remote file**

locking, 3-4 to 3-5

**remote file system**

displaying mounted, 3-2  
hard-mounted, 4-1  
mounting, 3-1 to 3-3  
mounting procedure, 4-3  
mounting with hard option, 3-2  
mounting with soft option, 3-2  
soft-mounted, 4-1  
troubleshooting slow access, 4-9

**remote superuser**

access privileges, 2-2n, 3-6  
allowing NFS access, 3-7, 3-7n,  
3-11

**root**

*See* remote superuser

**rpcinfo command**

NFS and, 4-2

**rwalld daemon**

starting, 2-6

**S**

**sendmail daemon**

restarting, 3-8  
terminating, 3-8

**server**

*See also* client  
client and, 1-2  
defined, 1-1

**system**

hanging during boot, 4-8

**system security**

exporting directories and, 3-10  
exporting file systems and, 3-10  
improving, 3-10 to 3-11  
NFS and, 3-6 to 3-11

**W**

**wait lock**

*See lock*

**write lock**

*See lock*

**Y**

**YP service**

hanging system and, 4-8





## HOW TO ORDER ADDITIONAL DOCUMENTATION

### DIRECT TELEPHONE ORDERS

In Continental USA  
and New Hampshire,  
Alaska or Hawaii  
call **800-DIGITAL**

In Canada  
call **800-267-6215**

### DIRECT MAIL ORDERS (U.S. and Puerto Rico\*)

DIGITAL EQUIPMENT CORPORATION  
P.O. Box CS2008  
Nashua, New Hampshire 03061

### DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.  
100 Herzberg Road  
Kanata, Ontario K2K 2A6  
Attn: Direct Order Desk

### INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION  
PSG Business Manager  
c/o Digital's local subsidiary  
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminister, Massachusetts 01473

\*Any prepaid order from Puerto Rico must be placed  
with the Local Digital Subsidiary:  
809-754-7575



## Reader's Comments

**Note:** This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Did you find errors in this manual? If so, specify the error and the page number. \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) \_\_\_\_\_

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_

Street \_\_\_\_\_

--- Do Not Tear - Fold Here and Tape ---

**digital**



No Postage  
Necessary  
if Mailed in the  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation  
Documentation Manager  
ULTRIX Documentation Group  
ZKO3-3/X18  
Spit Brook Road  
Nashua, N.H.

03063



--- Do Not Tear - Fold Here and Tape ---

Cut Along Dotted Line