ULTRIX

digital

Guide to the Network File System

Order Number: AA–ME99B–TE

ULTRIX

Guide to the Network File System

Order Number: AA-ME99B-TE
June 1990

Product Version: ULTRIX, Version 4.0 or higher

The Network File System allows users to mount directories across the network and treat those directories as if they were local. This guide describes the Network File System.

digital equipment corporation
maynard, massachusetts

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause of DFARS 252.227-7013.

© Digital Equipment Corporation 1990
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:

digital	DECUS	ULTRIX Worksystem Software
	DECwindows	UNIBUS
CDA	DTIF	VAX
DDIF	MASSBUS	VAXstation
DDIS	MicroVAX	VMS
DEC	Q-bus	VMS/ULTRIX Connection
DECnet	ULTRIX	VT
DECstation	ULTRIX Mail Connection	XUI

Ethernet is a registered trademark of Xerox Corporation.

Network File System and NFS are trademarks of Sun Microsystems, Inc.

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

Contents

About This Guide

The Network File System allows users to mount directories across the network and treat those directories as if they were local. This guide describes the Network File System.

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 VAX or RISC 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 appendix:

Chapter 1	Introduces the Network File System (NFS) and describes basic NFS concepts. This chapter also provides the background information needed before you can set up and run NFS on your system.
Chapter 2	Describes how to set up NFS manually, including how to set up the automount program on a client machine. The description is included for those who want to understand how NFS operates and which files are affected by NFS.
Chapter 3	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 also 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	Describes the basic approach to solving NFS-related system problems. This chapter also discusses various system problems you may encounter and describes how to solve them.

Appendix A	Lists remote mount error messges, automount error messages, NFS console 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:

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.
#	A number sign is the default superuser prompt.
`user input`	This bold typeface is used in interactive examples to indicate typed user input.
`system output`	This typeface is used in interactive examples to indicate system output and also in code examples and other screen displays. In text, this typeface is used to indicate the exact name of a command, option, partition, pathname, directory, or file.
UPPERCASE lowercase	The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
rlogin	In syntax descriptions and function definitions, this typeface is used to indicate terms that you must type exactly as shown.
[ ]	In syntax descriptions and function definitions, brackets indicate items that are optional.
{ | }	In syntax descriptions and function definitions, braces enclose lists from which one item must be chosen. Vertical bars are used to separate items.
. . .	In syntax descriptions and function definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
`cat(1)`	Cross-references to the *ULTRIX Reference Pages* include the appropriate section number in parentheses. For example, a reference to `cat(1)` indicates that you can find the material on the `cat` command in Section 1 of the reference pages.

New and Changed Information

This manual is a revision; it includes the following new information:

- Using the automount command (Chapter 2)
- Automount error messages (Chapter 4)

In addition, there are editorial, formatting and typographical changes.

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.

The following topics are discussed in this chapter:

- The NFS environment
- The NFS service
- The NFS locking service

1.1 The NFS Environment

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 on a local system and then reading or writing the files as though they were local.

Sharing file systems:

- Removes the need to copy files across the network from one system to another
- Provides easier access to remote files
- Reduces the risk of having out-of-date copies of the same file on different systems

Client systems request resources provided by other systems, called servers. A server is any host system or process that provides a network service. A client is any host system or process that uses services from a server.

A single host can provide more than one service. Servers are passive; they wait for clients to call them. Servers never call the clients.

A one-to-one correspondence between servers, clients, and systems does not always exist. A system that acts as a server can also act as a client. 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. Table 1-1 summarizes the distinctions between client and server systems for NFS mounts.

Table 1-1: Comparison of Client and Server Relationships

Client	Server
Requests a remote mount	Responds to mount request
Reads `/etc/fstab` file	Reads `/etc/exports` file
Checks if server is known	Checks if client is known

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.

You can mount a remote file system by using either the `mount` command or the `automount` command. For more information, see Chapter 2.

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.

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

1.2 The NFS Service

This section describes the service that NFS provides, including the `/etc/fstab` and `/etc/exports` files and 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. Once a remote file system or directory is mounted, it can be used just as a local file system by client programs. Typically, a client mounts one or more remote file systems and directories at startup time, if lines similar to the following example 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:
```

The first line in the example shows that the directory `/usr2` at system `titan` is mounted at local mount point `/usr2` when you boot the local system. The rest of this line describes the mount. 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.

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. Advisory locks are not enforced.

Note

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

File locking is a way to manage shared file access. A process takes the following steps when locking a file or region of a file:

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 kernel and status monitor of the local system, as well as with the other 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 server 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 by using 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 the Network File System (NFS) in the following ways:

- Manually by editing `/etc/rc.local` and `/etc/fstab`
- Using the `automount` program

For information on setting up NFS automatically by using the `nfssetup` command, see the *Guide to System and Network Setup.* The following topics are discussed in this chapter:

- Preparing to set up NFS
- Setting up an NFS server manually
- Setting up an NFS client manually using `/etc/fstab`
- Setting up an NFS client using the `automount` program
- Modifying the NFS environment

2.1 Preparing to Set Up NFS

For NFS to run correctly and for remote mounts to work, both the client and the server must be established on a local area network. You also must know the answers to the following questions:

- Will your system act as a server? That is, will your system export file systems and directories?

  If your system is to be a server, then you must have the following information:

  - How much of an NFS workload will your system get?

    For an average workload, four `nfsd` daemons and four `biod` daemons (block input/output) are sufficient. You can configure more or less, depending upon the workload you anticipate.

  - What are the absolute pathnames of the file systems and directories your system will export? The following is an example of an absolute 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, do not allow a remote superuser access to your system unless the remote hosts (and superusers) on your network are trusted.

- Will your system act as a client? That is, will your system access file systems and directories exported by other hosts (servers) running NFS on your network?

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

  - What are the names of remote hosts whose file systems and directories you will import (remote mount on your local system)?
  - What are the absolute pathnames of the remote file systems and directories you will mount from the remote host (server)?
  - What are the absolute pathnames of the local mount points on your system where you will mount the remote file systems and directories? A common convention is to create local mount points whose pathnames correspond to the pathnames on the remote systems. You can, however, create local mount points whose pathnames differ completely from the remote pathnames.

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. If you set up NFS using the `nfssetup` command, it checks your kernel to verify that NFS is configured properly and informs you if it finds configuration problems.

If you set up NFS manually and are uncertain whether it is configured into your kernel, check your system configuration file for the following line:

```
options         NFS
```

If NFS is not configured into your kernel, you must add it to the system configuration file and rebuild your kernel before attempting to set up NFS. See `doconfig`(8) in the *ULTRIX Reference Pages* for information on rebuilding your kernel.

2.2 Setting Up an NFS Server Manually

To set up an NFS server manually to export file systems and directories, follow these steps:

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

2.2.1 Edit the /etc/exports File

Add one entry per mount-point pathname for 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. For example, to export the file system `/usr/src/mybin` to the world, with no special permissions, the `/etc/exports` file would have an entry like this:

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

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

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

To deny NFS access to a file system or directory, remove its entry from the `/etc/exports` file.

Note

Whenever you make any modifications to the `/etc/exports` file, be sure to enter the following command:

```
# showmount -e
```

Entering this command ensures that the changes take effect immediately. For example, if you remove an entry from the `/etc/exports` file, entering this command ensures that the removal takes immediate effect. Likewise, if an exported filesystem is mounted, or re-mounted on another device, then the above command must be entered.

The server checks each NFS request 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. The `/etc/exports` file can contain only one entry forr 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
/usr/projects/vm  -r=0  blue white   # export directory to hosts blue
                                     # and white - allow clients blue
                                     # and white to have superuser
                                     # access to this directory
```

In this example, 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 only permissions, 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`. The `-r=0` option in the final line of the example allows client superusers access to the `/usr/projects/vm` directory with the same permissions as a local superuser.

Note

The pathnames specified in the `/etc/exports` file can be any local directories and are not limited to file system mount points. However, exports do not cross file system boundaries. For example, because `/` and `/usr` are separate file systems on the server, 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.

2.2.2 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
```

2.2.2.1 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.

2.2.2.2 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 -i ; echo -n ' mountd -i'  > /dev/console
fi
```

If this entry does not exist, add it, or a similar one, to the `/etc/rc.local` file.

The `mountd` entry must come after the `portmap` entry in the `/etc/rc.local` file. See `mountd`(8nfs) in the *ULTRIX Reference Pages* for further information.

2.2.2.3 Make Entries for the NFS Daemons

Make sure that the `nfsd` server daemons are available to provide the NFS service. A typical number of `nfsd` 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 an entry like this:

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

If this entry does not exist, add it, or a similar one, to the `/etc/rc.local` file.

The `nfsd` entry must come after the `mountd` entry in the `/etc/rc.local` file.

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

2.2.2.4 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 `nfssetup` command as described in the *Guide to System and Network Setup*. 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.

2.2.3 Reboot the System

After you have edited the `/etc/rc.local` and `/etc/fstab` files (if you are using the `/etc/fstab` to mount file systems), 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 commands and daemons, but not NFS locking-related, without having to bring the system to single-user and then multiuser mode, enter the following commands as superuser:

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

Caution

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.

2.3 Setting Up an NFS Client Manually Using /etc/fstab

To set up your system as an NFS client using the `/etc/fstab` file to import file systems and directories, do the following:

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

   See Section 2.2.3 for information on how to reboot the system.

2.3.1 Edit the /etc/rc.local file

NFS clients must run the `portmap` and `biod` daemons. If you want your system to be notified when an NFS server is going down, you must also run the `rwalld` daemon.

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
```

2.3.1.1 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.

2.3.1.2 Make an Entry for the Block Input/Output Daemon – Make sure that the `biod` client daemon is available to provide the NFS service. A typical number of `nfsd` 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 an entry like this:

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

If this entry does not exist, add it, or a similar one, to the `/etc/rc.local` file. The `biod` entry must come after the `portmap` entry.

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

2.3.1.3 Make Entry for the rwalld Daemon – 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
```

The `rwalld` entry must come after the `portmap` entry.

2.3.1.4 Make 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 `nfssetup` command as described in the *Guide to System and Network Setup*. 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.

2.3.2 Mount File Systems Using /etc/fstab

Use the `/etc/fstab` to mount file systems you always want to have mounted. 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/src` to be mounted automatically from an NFS server named `spice` onto the local mount point `/spice/usr/src`, place an entry similar to the following in the `/etc/fstab` file:

```
/usr/src@spice:/spice/usr/src:ro:0:0:nfs:bg:
```

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 -o bg,ro spice:/usr/src /spice/usr/src
```

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

2.4 Setting Up an NFS Client Using the Automount Command

When you find that certain file systems are used occasionally rather than frequently, the automount command is an alternative to the /etc/fstab file for mounting file systems on client machines. The automount command forks an automount daemon that is booted when the /etc/rc.local file is read at startup. The daemon listens on a specified port, intercepts NFS requests to the kernel, and automatically mounts and unmounts the file systems as they are needed. The automount command can be entered from the command line or it can be placed in the /etc/rc.local file.

Note

Specify remote file systems by using either the automount command or the /etc/fstab file, but not both. If a particular file system appears in both an automount map and the /etc/fstab file, you get the following error message:

hostname: *file system* already mounted on *mountpoint*

A user defines a series of maps that specify one of the following:

- The remote file systems that the automount program will mount and unmount
- Pointers to other maps that contain the necessary information

  The names of the maps can be passed to the automount program from the command line or from a master map. The maps can be local or served by the Yellow Pages Service.

When a user requests access to a remote file hierarchy associated with an automount map, the automount daemon does the following:

1. Intercepts the kernel NFS request.
2. Consults the appropriate map for the location of the file system. If is no such map exists, the automount program looks for a Yellow Pages map by that name.
3. Mounts the file system under the /tmp_mnt directory
4. Creates a symbolic link between the requested mount point and the actual mount point in the /tmp_mnt directory.
5. Automatically unmounts the file system after a period of inactivity (5 minutes by default).

Table 2-1 compares use of the `mount` and `automount` commands.

Table 2-1: Comparison of Mount and Automount

mount	automount
Consults `/etc/fstab` file for the list of remote file systems when used with `-a` option	Consults specified maps for the list of remote file systems and checks for the existence of YP map named `auto.master`
Mounts directories at boot time if called from `rc.local`	Mounts directories only when accessed by user on client system
Directories remain mounted	Directories mounted only when in use
Directories must be unmounted by using `umount` command	Directories unmounted automatically if not used for specified time period
The `/etc/nfssetup` command adds `mount` to `/etc/rc.local`	The `automount` command must be added manually to `/etc/rc.local`
Can be called from the command line	Can be called from the command line

A server does not know whether the files it exports are accessed through `mount` or `automount`. Therefore, you do not have to do anything different on the server to use the `automount` program on your clients.

There are three types of `automount` maps:

- master
- direct
- indirect

Errors detected by the `automount` program are issue to the screen or console, or are written to `syslog`. See Appendix A for a listing of automount error messages.

2.4.1 The Master Map

At startup, the `automount` program checks for the presence of a YP map named `auto.master`. You are not required to run the YP service or have a distributed `auto.master` map to use the `automount` program. You can also specify a local master file by using the `-f` option on the command line. For information on invoking the `automount` program, see Section 2.4.9.

If you make use of both a distributed `auto.master` map and a local master map, entries in the local map take precedence and will override conflicting entries in the `auto.master` map.

Entries in a master map point to other maps that contain information about the location of remote file systems and where they should be mounted locally.

Each line in the `/etc/auto.master` file has the following syntax:

```
Mount-point     Map   [Mount-options]
```

- *Mount-point* is the absolute pathname of a directory for indirect maps. The mount point for direct maps must always be `/-` in a master map.
- *Map* is the name of the map the `automount` program consults for the names of remote file systems and their local mount points.
- *Mount-options* is a list of options used to regulate the mounting of entries listed in *map*. The list is comma separated and is optional. If *map* entries specify options other than those listed in *mount-options* those listed in the *map* entry take precedence.

A typical `/etc/auto.master` file looks like the following:

```
#
#Mount-point          Map                   Mount-options
#
/net                  -hosts
/home                 auto.home             -rw,intr
/-                    /etc/auto.direct      -ro,intr
```

The number sign (#) is the comment character in map files.

The `/net` directory is the local mount point where the `automount` program mounts all of the entries in the `-hosts` map. The `-hosts` map is a special map that uses the hosts database `/etc/hosts` or the Yellow Pages map `hosts.byname`. The `-hosts` map gives access to the exported NFS file systems of any server on the local area network. For more information on the `-hosts` map, see Section 2.4.6.1.

The `/home` mount point is the directory under which the entries listed in the YP map `auto.home` are mounted. They are mounted under `/tmp_mnt/home/`*directory* with a symbolic link to `/home/`*directory*. Note that the names `/net` and `/home` are arbitrary names. The `automount` program creates them if they do not exist.

The `/-` mount-point is a dummy directory that the `automount` program recognizes as the specification of a direct map. Unlike an indirect map or a special map, a direct map is not associated with any mount point. The direct map itself contains all the information needed to locate and mount remote file systems.

2.4.2 Direct Maps

Direct maps specify which remote file systems to mount locally and what the local mount points are. They do not point to other maps. You can also specify mount options.

The syntax of entries in a direct map is the same as the syntax in indirect maps:

```
Key  [mount-options]      location
```

- *Key* is the pathname of the mount point. The *key* field in a direct map is an absolute pathname.
- *Mount-options* are the options you want to apply to this particular mount. The mount-options need only be supplied if they differ from those given as a default for the whole map. If present, they override the mount options specified in the command line or in the master map.

All of the mount options available with the `mount` command are available with the `automount` command, except the `bg` and `fg` options, which do not apply. See `mount`(8nfs) in the *ULTRIX Reference Pages* for more information.

- *Location* is the location of the resource, specified as follows:

  server:pathname .

Note

The `automount` program cannot parse pathnames that contain colons. If the pathname contains a colon, use a backslash (\) to escape the special meaning of the colon.

If a directory name contains a space, place double quotation marks (" ") around the space.

The following is a typical entry in a direct map:

```
/usr/local/tmp   -ro    dancer:/usr/local
```

In the preceding example, `/usr/local/tmp` is the key, that is, the absolute pathname of the mount point on the client system: `-ro` refers to the options for the mount, and `dancer:/usr/local` is the name of the remote server and the directory to be mounted.

2.4.3 Indirect Maps

Indirect maps specify which remote file systems you want mounted locally and what the local mount points are. You can also specify mount options. The only difference between a direct and an indirect map is that the key in a direct map is an absolute pathname, where the key in an indirect map is a simple name that does not begin with a slash.

An indirect map can be specified in the master map with an entry similar to the following:

```
/home      /etc/auto.home      -rw,intr
```

In the preceding example, `/etc/auto.home` is the name of the indirect map that contains the entries to be mounted under the `/home` mount point on the client host.

Each line in an indirect map has the following syntax:

```
Key  [mount-options]      location
```

- *Key* is the name of the directory that will be used as the mount point. The *key* field in an indirect map is a simple pathname.
- *Mount-options* are the options you want to apply to this particular mount. If present, they override the mount options specified in the command line or in the master map.

  All of the mount options available with the `mount` command are available with the `automount` program, except the `bg` and `fg` options, which do not apply.
- *Location* is the location of the resource, specified as *server:pathname*

Note

The automount program cannot parse pathnames that contain colons. If the pathname contains a colon, use a backslash (\) to escape the special meaning of the colon.

If a directory name contains a space, place double quotation marks (" ") around the space.

Given this entry in the master map,

```
/home       /etc/auto.home      -rw
```

an example of an entry in the /etc/auto.home map is,

```
max         -rw,nosuid        host1:/home/host1/max
```

If auto.home had been specified as a direct map in the master map, for example,

```
/-     /etc/auto.home     -rw
```

then the following entry in /etc/auto.home would have the same results as the preceding indirect map example:

```
/home/max      -rw,nosuid       host1:/home/host1/max
```

Individual user's directories can be included in an automount map. For example, one of the entries in a master map might be the following:

```
/home            /etc/auto.home     -rw,intr
```

In this example, /etc/auto.home is the name of an indirect map containing the entries to mount under the /home directory. A typical auto.home map might look like the following:

```
#key       mount-options      location
willow                        willow:/home/willow
cypress                       cypress:/home/cypress
poplar                        poplar:/home/poplar
pine                          pine:/export/pine
apple                         apple:/export/home
ivy                           ivy:/home/ivy
peach       -rw,nosuid       peach:/export/home
```

Assume, for example, that the map in the preceding example is on host oak. If user ramone has an entry in the password database specifying his home directory as /home/willow/ramone, then whenever he logs into host oak, the automount program will mount (as /tmp_mnt/home/willow) the directory /home/willow residing on machine willow. If one of the directories is actually ramone, he will be in his home directory, which is mounted read/write and interruptable.

If, for example, the home directory for ramone is specified as /home/peach/ramone, whenever he logs into host oak, the automount program mounts the directory /export/home from host peach under /tmp_mnt/home/peach. His home directory will be mounted read/write, nosuid.

Note

Any option in the file entry overrides all options in the master map or from the command line.

Assume that the following conditions occur:

- The home directory for user `ramone` is listed in the password database as `/home/willow/ramone`.
- Host `willow` shares its `home` hierarchy with the hosts in `auto.home`.
- All those hosts have a copy of the same `auto.home` map and the same password database.

Under these conditions, user `ramone` can run `login` or `rlogin` on any of those hosts and have his home directory mounted in place for him. In addition, user `ramone` can also enter the following command:

```
% cd ~joseph
```

The `automount` program will mount the home directory of user `joseph` providing all permissions apply.

2.4.4 Hierarchical Mounts

You can mount different directories within a file system hierarchy from different servers. For example, if you are mounting the `/usr/local` file system on your machine, you can mount the various subdirectories within `/usr/local` from different servers.

In the following example, the directories `/usr/local/bin`, `/usr/local/src`, and `/usr/local/tools` are mounted from the machines `host1`, `host2`, and `host3` respectively.

```
/usr/local\
          /bin    -ro,soft        host1:/usr/local/bin \
          /src    -ro,soft        host2:/usr/local/src \
          /tools  -ro,soft        host3:/usr/local/tools
```

In the preceding example, the local key, `/usr/local`, comprises three subdirectories, each of which is a mount point for a remote directory on a different remote server. The example is displayed showing the entry split into four lines with the continuation lines indented for readability.

The preceding example shows multiple-nonhierarchical mounts under `/usr/local`. The following example shows a true hierarchical entry:

```
/usr/local \

             /          -ro,soft        host0:/usr/local \
             /bin       -ro,soft        host1:/usr/local/bin \
             /src                       host2:/usr/local/src \
             /tools                     host3:/usr/local/tools
```

The mount points used here for the hierarchy are `/`, `/bin`, `/src`, and `/tools`. Note that these mount points are relative to `/usr/local`. The mount point `/` mounts `/usr/local` from host0.

When file systems are mounted hierarchically, each file system is mounted on a subdirectory within another file system. When the root of the hierarchy is referenced, the `automount` daemon mounts the entire hierarchy.

2.4.5 Replicated File Systems

You can specify multiple locations for a single mount. If a file system is located on several servers and one of them goes down, the file system can be mounted from one of the other servers. This makes sense only when mounting a read-only file system.

In the following example the Reference Pages can be mounted from `host1`, `machine2`, or `system3`.

```
/usr/man\
          -ro,soft  host1:/usr/man \
                    machine2:/usr/man \
                    system3:/usr/man
```

The preceding example can also be expressed as a list of servers separated by commas and followed by a colon and the pathname, for example:

```
/usr/man  -ro,soft  host1,machine2,system3:/usr/man
```

This syntax is valid only if the pathname is the same on each server.

When you access the reference pages, the `automount` program issues a `ping` command to each of the specified servers. The server that answers the `ping` first is used for the mount.

If a server goes down while a client has a file system mounted, the file system becomes unavailable. If this happens, you should wait for the `automount` program's time-out period to expire. After the time-out period expires, the `automount` program unmounts the file system. When you try to access the file system again one of the other servers responds to the request and the file system is mounted, but from a different server.

Another option is to use the `umount` command, inform the `automount` daemon of the change to the system mount table (as specified in Section 2.4.10.3), and retry the `mount`.

2.4.6 Built-In Maps

The `automount` program provides the following built-in maps:

- The `-hosts` map
- The `-passwd` map
- The `-null` map

The following sections discuss these maps.

2.4.6.1 The –hosts Map

The –hosts Map – The `-hosts` map gives access to the exported NFS file systems of any server in `/etc/hosts`, or in the YP map `hosts.byname`. For example, with the `-hosts` map mounted on the `/net` mount point, enter the following command:

```
% ls /net/gumbo
```

This command mounts all exported file systems from host `gumbo` that this client is allowed to mount.

The actions of the `automount` program with respect to the command in the preceding example are as follows:

1. Contacts the server's mount service and requests a list of the exported file systems
2. Sorts the list according to length of pathnames to ensure the correct mounting order
3. Builds an internal hierarchical map
4. Proceeds down the list, mounting all the file systems in the `/tmp_mnt` directory and creates mount points as needed
5. Returns a symbolic link that points to the top of the mounted hierarchy

Note that the `automount` program must mount all the file systems that the indicated server advertises for sharing. Create a specialized map if frequent access to a specific file system is required.

2.4.6.2 The –passwd Map

The –passwd Map – The `-passwd` map is included to provide easy access to users' home directories around the network. It works only for a specified format of home directory paths. The format is as follows:

```
/name/server/loginname
```

Given the login name of a user, the `automount` program makes a call to `getpwnam ( )` to get the user's home directory path. It uses the server name in the middle component of the path to build an internal map entry. For example, given the following entry in the master map:

```
/net     -passwd      -rw
```

A user enters the following command:

```
% cd /net/max .
```

The `automount` program builds the following internal map entry:

```
max          -rw          server:/home/server/max
```

It then mounts the directory and changes the user's current directory to `/tmp_mnt/net/max`.

2.4.6.3 The –null Map

The –null Map – The `-null` map, when indicated on the command line, cancels the map associated with the indicated directory. It can be used to cancel a map specified in the master map.

For example, invoking the `automount` program with the following command causes the `/net` entry in `auto.master` to be ignored:

```
# automount /net -null
```

2.4.7 Substitutions and Pattern Matching

The `automount` program recognizes the following substitution characters, thus allowing you to eliminate redundancy within automount maps:

- ampersand (&)
- asterisk (*)

Recall that lines in direct and indirect maps have the following syntax:

```
key        mount-options        location
```

Whenever the `automount` program encounters an ampersand (&) in a line of a direct or indirect map, it substitutes the *key* in that line for the ampersand (&).

The following example is an indirect map that is not using ampersands:

```
#key              mount-options       location
#
host1             -rw,nosuid          host1:/home/host1
host2             -rw,nosuid          host2:/home/host2
```

Using the ampersand (&) as a substitution character, the entries read as follows:

```
#key              mount-options       location
#
host1             -rw,nosuid          &:/home/&
host2             -rw,nosuid          &:/home/&
```

Use the asterisk (*) to substitute for lines that are all formatted similarly. The `automount` program uses the asterisk to match any host not listed as a key in an entry before the asterisk. A typical line reads as follows:

```
#key       mount-options        location
#
*          -rw,nosuid              &:/home/&
```

Note

The `automount` program ignores any entry in an indirect map that follows an asterisk.

2.4.8 Environment Variables

You can use the value of an environment variable in a map by prefixing its name with a dollar sign ($). You also can use braces ({ }) to delimit the name of the variable from appended letters or digits.

The environment variables can be inherited from the environment or can be defined explicitly with the `-D` option on the command line. For example, you can invoke the `automount` program with the `HOST` variable by entering the following command:

```
# automount -D HOST=`hostname`
```

The following is an example of a direct map entry that uses the environment variable `HOST`:

```
/mydir     -rw      server:/export/$HOST
```

2.4.9 Invoking the Automount Program

The `nfssetup` command does not put `automount` command startup lines in `/etc/rc.local`. You must manually add the lines to start the `automount` program, preferably after the lines for NFS start and end. The following example provides a sample of the lines you add to `/etc/rc.local`:

```
# %AUTOMOUNT% - for starting the automount program
echo -n 'automount daemons;'
[ -f /usr/etc/automount ] && {
     /usr/etc/automount -f /etc/auto.master;
}
```

The syntax for invoking the `automount` program from the command line is as follows:

# **automount [-mnTv] [-f***master-file*] [-M*mount-directory]*\
[-tsub-options] [*directory map [-mount-options]]* ...

The options are explained in `automount`(8nfs) in the *ULTRIX Reference Pages.*

You can invoke the `automount` program from the command line or from an entry in the `/etc/rc.local` file in one of the following ways:

1. Specify all of the arguments to the `automount` command without referring to a local master map.

   The following line invokes all arguments to the `automount` command without reference to a local master map:

   ```
   automount /net -hosts /home /etc auto.home -rw,intr \
        /- /etc/auto.direct -ro,intr
   ```

2. Include the above information in a YP distributed `auto.master` map.

   Assuming that a YP `auto.master` map did not exist in the preceding example, the following command produces the equivalent results:

   ```
   automount
   ```

3. Include the information in steps 1 and 2 in a local `auto.master` file, and instruct the `automount` program to consult it for instructions.

   ```
   automount -f /etc/auto.master
   ```

4. Specify mount points on the command line, in addition to those included in the `auto.master` file. For example:

   ```
   automount -f /etc/auto.master /src /etc/auto.src -ro,soft
   ```

5. Nullify one of the entries in the master map. For example:

   ```
   automount -f /etc/auto.master /home -null
   ```

6. Replace an entry in the `auto.master` map with one of your own. For example:

   ```
   automount -f /etc/auto.master /home /mine/auto.home -rw,intr
   ```

Shell filename expansion does not apply to objects that are not currently mounted. For example, a `readdir`, such as `ls` or `find`, of an indirect automount mount point, for example, `/home`, will return only the currently active symbolic links; it will not enumerate the map.

2.4.9.1 The Temporary Mount Point – By default the automount prgram mounts remote file systems under the /tmp_mnt directory.

You can change the default mount point using the -M option. If you specify a different directory in either the rc.local file or on the command line, the automount program mounts directories at the new location. The following command indicates that the default temporary mount point should be /auto:

```
automount -M /auto
```

2.4.10 Administration of Automounted Systems

You can administer the automount command, its mount points, and its maps on a single host by using regular files or across an entire YP domain by using YP maps. The rest of this section discusses administration.

2.4.10.1 Yellow Pages Administration – The Yellow Pages service is not required to use the automount program. If you are running YP, the automount program checks for the presence of a YP map called auto.master at startup. To prevent the automount program from consulting the auto.master map, use the -m option with the automount command.

When given a map name, the automount program first checks for a local file. If such a file does not exist, the program assumes that the name refers to a YP map.

Changes made to a YP map will affect every client using the automount program in the YP domain. This simplifies the task of moving file systems from one server to another. Changes in the locations of file systems need only be made in the appropriate YP maps. These changes are transparent to clients who no longer need to be aware of the location of a remote file system and no longer need to update their /etc/fstab files when an NFS file system is moved.

A file map entry can be a reference to a YP map. If, in the course of scanning a local map for a key, the automount program finds a key with a prepended plus sign (+), the program treats the key as the name of a YP map to consult. This feature may be used to interpose map entries that are specific to the host before the shared YP map is consulted.

The map in the following example provides access to various source trees. If the key is not bsd4.2, the YP map auto.src is consulted for the key. If it is not found in auto.src, the key is caught by the "catchall" entry, which assumes that the key is a server name. The automount program then attempts to mount /usr/src from that server. For example:

```
bsd4.2              host:/usr/arch/src/bsd4.2
+auto.src
*                   &:/usr/src
```

2.4.10.2 Changing Local and YP Automount Maps – Master maps are only read when the automount program is started. Changes made to master maps do not take effect until the automount program is restarted. The maps you set at startup are fixed; you cannot add or disassociate a direct or indirect map after startup.

You can, however, modify the mount options and location fields of existing entries in direct and indirect maps. For example, you edit the /etc/auto.direct file so that the directory /usr/src is now mounted from a different server. The modified entry takes effect immediately if /usr/src is not mounted. If it is mounted, you can either wait until the auto-unmount takes place or unmount it manually, informing the `automount` program of the change in the mount table. For more information on unmounting an automounted file system, see Section 2.4.10.3.

You can add entries to an indirect map. The `automount` daemon monitors attempts to access the mount point associate with that map and scans the map every time the mount point is referenced.

However, the `automount` daemon monitors attempts to access each direct map entry individually. When the absolute pathname for the direct map entry is accessed, the `automount` daemon wakes up and reads the entry. Therefore, you can add an entry to, or delete an entry from, a direct map, but the change will not take effect until the `automount` program is restarted.

2.4.10.3 Unmounting File Systems Manually – The `automount` program needs to keep track of the file systems it has mounted. If you use the `umount` command to unmount one of the automounted hierarchies, a directory under `/tmp_mnt`, the `automount` program must be informed. Sending a `SIGHUP` signal to the `automount` program instructs the program to read the system's mount table. By reading the mount table, the `automount` program can identify which automounted file systems have been unmounted.

To notify the `automount` daemon, obtain its PID by entering the following command and observing the output as shown in this example:

```
# ps -x | grep automount
 229 p0 I    0:02  automount /home
```

The command in the preceding example gives the PID of 229. You then can send the `automount` daemon the `SIGHUP` signal with the following command:

```
# kill -1  229
```

2.5 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.

2.5.1 Modifying the NFS Environment Manually

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 2.2.1. 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 2.3.2.

2.5.2 Modifying the NFS Environment Automatically

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. However, you can enable or disable it manually by editing the `/etc/rc.local` file as described in Section 2.2.2.

Managing the Network File System 3

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

- 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 can mount a remote file system or directory onto a local mount point, provided the following conditions are met:

- It can reach the NFS server over the network
- 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 the following:

```
# 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. Since not all applications properly test return codes, as a general rule, it is best not to use the soft option with file systems mounted as read-write.

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.

The `interrupt` option is the default with the `hard` option. When you use the interrupt option, `intr`, with the `hard` option, you can type an interrupt character and prevent your system from indefinitely attempting to reach an unreachable server system. 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, for example:

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

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` on the remote host `spice` to be mounted automatically at startup time on the local system on `/spice`:

```
/usr@spice:/spice:rw:0:0:nfs:bg:
```

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.

The following example illustrates a `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. 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` primatives 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 following list provides the key elements for locking a remote file:

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

The following list 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.

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 affected 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 in the previous list.

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 following topics are discussed in this section:

- 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, that is, local file systems, 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 thus denied access to remotely mounted files and directories whose permissions do not allow world access.

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, do 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 the server's `/usr/games` file system as the superuser on the server:

```
/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. The address all mail intended for the administrators of that system as follows:

```
        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 this:

   ```
   CN MAILER-DAEMON postmaster
   ```

   The line becomes this:

   ```
   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 -q1h -om
   ```

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

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. By default, 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 the following:

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

To turn off source port checking, 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, follow these steps:

1. Delete the `/etc/exports` file
2. Enter the following command to ensure that the deletion takes immediate effect:

   ```
   # showmount -e
   ```

3. 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 2 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 entire file system `/usr/staff`, however, can be mounted remotely (read-only) only from the client `pink`. The subdirectory `/usr/staff/doe` can be mounted read-write remotely only from `green`. The directory `/usr2` can be mounted remotely only from hosts `pink`, `green`, and `black`.

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 `fsirand` 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 `fsirand` command, you must unmount the file system and then run the `fsck` command on it. For example, to use `fsirand` 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 must 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 protects 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.

Each NFS request is checked to ensure that the file system being accessed is currently exported. If the file system is not exported, the operation is rejected and the

ESTALE errno value is returned to the client process. An unexported file system message is also written to the server console and to the system error log.

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 discusses these peculiarities and offers suggestions on how to solve any 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 might 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 greater than six months' worth of seconds, the file is considered old.

  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:

```
(current time) - (modification time) =
(current time) - (current time + 61 minutes) = -61 minutes
```

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. See Appendix A for a list of the NFS error messages and suggested 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.

There are three main points of failure when NFS is not working: the server, the client, or the network itself. The troubleshooting strategies in the following sections show 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` option to the `mount` command. If you specify neither `hard` nor `soft` 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.

It is possible that a process that accesses a hard-mounted file will fail to complete execution or hang for an unusually long time. 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 the following:

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

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

```
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

This section describes 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 issued from node `salt`:

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

Here are the steps the `mount` command takes to mount a remote file system from server system `sugar` onto the client system `salt`. 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 file system or directory is not already mounted on that mount point.
2. The `mount` command parses the first argument into host `sugar` and the remote directory `/usr/src`.
3. If the Yellow Pages Service 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 the port mapper on `sugar` to get the port number of the `mountd` daemon on `sugar`.
5. The `mount` command calls the `mountd` daemon on `sugar` 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, the `mountd` daemon on `sugar` 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. The `mountd` daemon on host `sugar` 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. The `mountd` daemon on host `sugar` 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 Process Blocking in Client Programs

Client programs can block, fail to complete execution, or hang for unusually long periods 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

4.3.1 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 automatically continues 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.

When a server is down, operations on soft-mounted remote files from a down server will 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.

4.3.2 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.

4.3.3 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.

4.4 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). An `fstab` entry with the background option might appear as follows:

   ```
   /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.

4.5 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 `biod` daemons 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.

Appendix A

This appendix provides listings and suggested user actions for the following classes of error messages:

- Remote mount error messages
- Automount error messages
- NFS console error messages

A.1 Remote Mount Error Messages

The error messages in this section are arranged according to when during the `mount` procedure the failures occur. They also show most likely error message labels.

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 mount point or remote file system name, but not both, it looks in the `/etc/fstab` file for a matching field. 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 `nfsd` daemons on the server. 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 the following:

```
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 the following:

```
# /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) 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.

A.2 Automount Error Messages

The error messages in this section are issued to the screen or console or sent to syslog by the automount program. The messages and a brief explanation of the problem are listed.

no mount maps specified

The automount program cannot find any maps to serve, nor can it find any Yellow Pages maps either. This message is returned only when you specify the -v switch.

mapname: Not found

The automount program can not locate the map it requires. This message is returned only when you specify the -v switch.

dir *mountpoint* must start with '/'

The automount program mountpoint must be given as full pathname. Check both the spelling and pathname of the mount point.

mountpoint: Not a directory

The *mountpoint* exists but is not a directory. Check both the spelling and pathname of the mount point.

hierarchical mountpoint: *mountpoint*

The automount program will not allow itself to be mounted within an automounted directory. Use another strategy for mounting the directory.

WARNING: *mountpoint* not empty!

The mountpoint directory is not empty. This message is returned only when you specify the `-v` switch. It is warning you that the previous contents of *mountpoint* will not be accessible while the mount is in effect.

Can't mount *mountpoint*: *reason*

The automount program cannot mount itself at *mountpoint.* The *reason* should be self-explanatory.

hostname: *file system* already mounted on *mountpoint*

The `automount` program has been mounted on an already-mounted on mount point and is attempting to mount the same file system there.

WARNING: *hostname:file system* already mounted on mountpoint

The automount program is mounting itself on top of an existing mount point. This message is a warning only.

couldn't create directory: *reason*

The automount program could not create a directory. The *reason* should be self-explanatory.

bad entry in map *mapname*

map *mapname*, key *key*: bad

The *map entry* in *mapname* is malformed and the `automount` program cannot interpret it. Recheck the entry; perhaps there are characters in it that need escaping.

mapname: yp_err

The automount program encountered an error in looking up a Yellow Pages map entry.

hostname: exports: *rpc_err*

The `automount` program encountered an error attempting to get the list of exported file systems and directories that it is allowed to mount from *hostname.* This error occurs when a user attempts to access a mount point that has the –hosts map associated with it. This error indicates a server or network problem.

host *hostname* not responding

hostname:filesystem server not responding

Mount of *hostname:filesystem* on *mountpoint: reason*

The automount program attempted to mount from *hostname* but got no response or failed. These errors could indicate a server or network problem.

mountpoint - pathname from hostname: absolute symbolic link

The automount program detected that *mountpoint* is an absolute symbolic link (begins with /). The content of the link is *pathname.* Because this may have undesired consequences on the client, the `automount` program will not mount on absolute symbolic links.

The following errors indicate problems attempting to `ping` servers for a file system that is exported from multiple servers as specified in a multiple server map entry. This could indicate network problems.

Cannot create socket for broadcast rpc: *rpc_err*

Many_cast select problem: *rpc_err*

Cannot send broadcast packet: *rpc_err*

Cannot receive reply to many_cast: *rpc_err*

trymany: servers not responding: *reason*

No server in a multiple-server map entry is responding. This indicates that the replicated file system could not be reached on any of the specified servers. This could indicate network problems.

Remount *hostname:filesystem* on *mountpoint*: server not responding"

The `automount` program could not remount a file system it had just unmounted. The `automount` program was attempting to remount *filesystem* because it discovered that a part of the automounted hierarchy at *mountpoint* was busy. The remote file system's server, *hostname,* did not respond to the mount request. This error indicates a server problem.

NFS server (pid*n@mountpoint*) not responding still trying

An NFS request to the automount daemon with PID *n* serving *mountpoint* has timed out. The automount daemon may be overloaded or dead. If the condition persists, reboot the client, otherwise you will have to exit all processes that are using automounted directories, kill the current automount process, and restart it from the command line.

A.3 NFS Console Error Messages

Under certain circumstances an NFS server can receive simultaneous requests from NFS clients for unmounted, unexported, or deleted file systems. The resulting volume of console messages can make the console unusable. Output is limited to ten "not mounted" messages followed by a ten-minute interval when no messages are displayed. After this interval, ten more messages are displayed and so forth.

Similarly, after ten "stale file handle" messages have been displayed for a particular file system, no messages are displayed for a ten-minute interval.

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 server is 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

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	**Call**	**Contact**
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Guide to the Network File System
AA-ME99B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? ______________________

What do you like best about this manual? ______________________

What do you like least about this manual? ______________________

Please list errors you have found in this manual:

Page Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using? ______________

Name/Title ______________________ Dept. ______________

Company ______________________ Date ______________

Mailing Address ______________________

Email ______________________ Phone ______________

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
Guide to the Network File System
AA-ME99B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? ______________________________

What do you like best about this manual? ______________________________

What do you like least about this manual? ______________________________

Please list errors you have found in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using? ____________

Name/Title ______________________ Dept. __________

Company ______________________ Date __________

Mailing Address ______________________________

Email ______________________ Phone __________

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

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

------- Do Not Tear – Fold Here --

Cut
Along
Dotted
Line