English

Fujitsu Software BS2000

NFS

NFS users and NFS administrators

User Guide

Valid for:
NFS V3.0
POSIX A49
BS2000 V21.0B

Edition October 2023

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to: bs2000services@ts.fujitsu.com.

Certified documentation according to DIN EN ISO 9001:2015

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2015.

Copyright and Trademarks

Copyright © 2023 Fujitsu Technology Solutions GmbH.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Table of Contents

POSIX_NFS_en

1 Preface

NFS (Network File System) permits distributed file access in networks having different computer architectures and operating systems. The use of NFS presupposes a computer network in which communication can take place via TCP/IP.

1.1 Brief product description

The present NFS version for BS2000 corresponds to the NFS implementation contained in Reliant UNIX V5.44 and to the AT&T System V Release 4 variant.

NFS in BS2000 allows you to process data located on remote systems, and to make BS2000 data accessible to users on remote systems. Remote processors with a low hard disk capacity or PCs can thus take advantage of the considerably higher storage capacity and far more convenient and reliable data backup mechanisms of BS2000 systems.

In order to use NFS on a BS2000 computer, the *open*Net Server product with BCAM must be installed.

NFS V3.0 requires POSIX-BC.

1.2 Target group

The manual is intended for all NFS users. Use of this manual presupposes a knowledge of the UNIX and BS2000 operating systems and assumes that you have access to the manual "POSIX Basics".

1.3 Summary of contents

Chapters 2 and 3 provide an overview of NFS, how it is integrated in BS2000, and how to use NFS.

Chapter 4 contains a description of the commands, the daemons and the administration files.

Chapter 5 gives advice on troubleshooting and error recovery, along with measures you can undertake to enhance the performance of NFS.

1.4 Notational conventions

The following notational conventions are used in this manual:

In body text

italics

All syntax elements and also other file names, path names and commands are shown in *italics*.

> ¡ This symbol identifies important information and exceptional circumstances which you should be aware of.

In the syntax

`bold typewriter face`

Constants: These characters must be entered exactly as they are printed.

`normal typewriter face`

Variables: These characters are replaced by other characters which you select and enter.

`[ ]`

Optional: Everything which is enclosed in square brackets can be entered, but does not have to be. The square brackets themselves should not be entered unless this is expressly required.

`' '`

A blank which must be entered.

`...`

The preceding expression can be repeated. If blanks need to be entered between the repeated expressions, and the blank is not contained in the expression, a blank (`' '`) will precede the ellipsis symbol.

`|`

The vertical line separates alternative specifications.

In examples

`normal typewriter face`

Inputs and outputs. With character-oriented terminals input lines are concluded with the RETURN key, while with block-oriented terminals the same function is fulfilled by EM DUE, the key specifications are therefore omitted.

1.5 Changes compared to the previous manual

NFS V3.0 incorporates the following changes compared with the previous version:

- Removed obsolete or no longer valid sections:
  - README file
  - /etc/print - Templates for BS2000 print commands
  - Connecting a Windows PC
- Added information about access to bs2fs file systems via NFS.

2 Overview and integration in BS2000

This chapter provides an overview of NFS. It explains how NFS is integrated in BS2000 and which security mechanisms can be used for the protection of remote resources.

2.1 Overview of NFS

This section provides basic information on the functionality of NFS. First we will describe how to work with distributed file systems, and then we will explain the differences between the various NFS versions. You will learn how NFS executes read and write accesses to files, how file locks are implemented in NFS, and how to handle large file systems.

2.1.1 Working with distributed file systems

NFS allows local files and directories to be made available (synonyms: “released” or“shared”) for processing on a remote system, and files and directories made available by remote systems can be processed on the local system as if they were local ones. In Figure 1, for example, the directory V1 on the BS2000 computer can be processed on the UNIX computer as if it were part of the local file system. Of course, the reverse is also possible, whereby a file system from a remote system can be mounted and processed on the BS2000 computer.

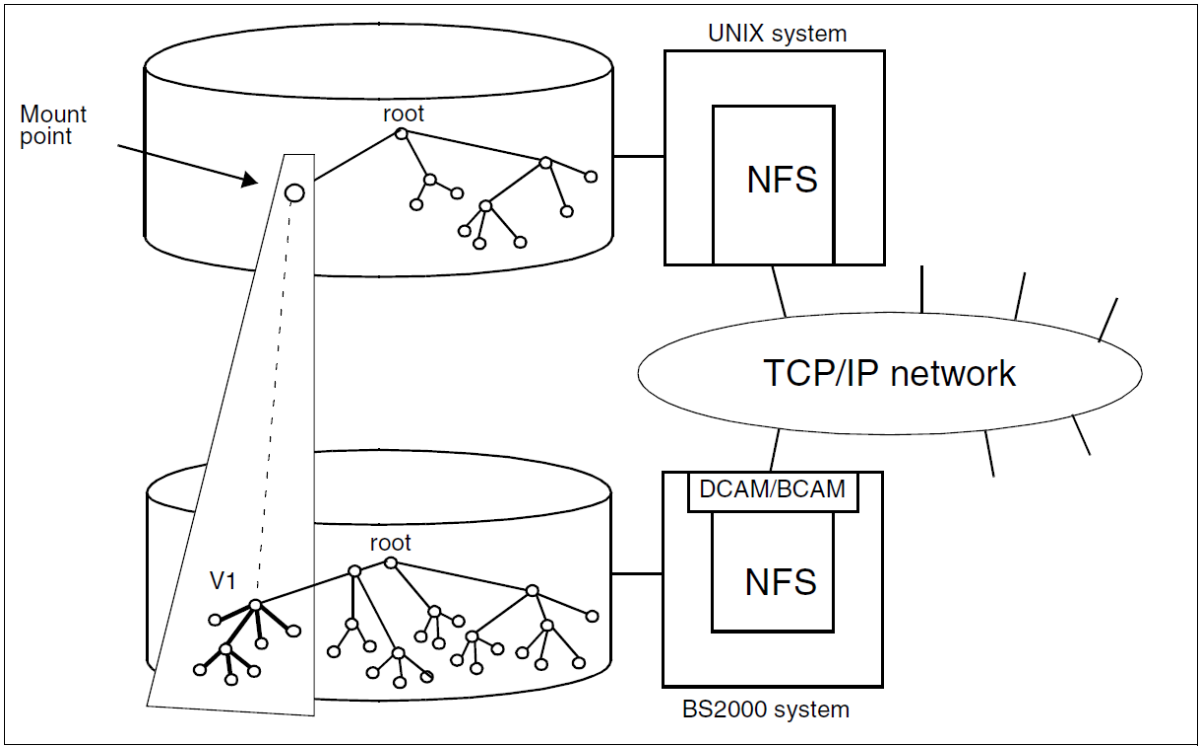

Figure 1: Distributed file access with NFS in a heterogeneous network

File system, directory and file

NFS operates with hierarchically organized file systems such as the UNIX file system which consists of directories and files. The entire file hierarchy of a UNIX computer comprises usually more than one file system. The files and directories of a single file system are physically located on the same storage medium, e.g. in a partition.

The files and directories of all file systems on a computer are organized in a tree structure. The root is the root directory. From the root, the structure branches into other directories. A directory is used to group files. The leaves are the files. No further branching is possible from the file level.

Resources

Since NFS defines an abstract file system model and operates across different operating systems, the term "resource" is used for all files and file hierarchies which are used with NFS. In UNIX environments resources are any sections of the file hierarchy, i.e. individual files, directories or a file system.

Client and server

NFS is based on the client-server principle:

- A system is called an NFS server if it makes available local resources for remote systems.
- A system is called an NFS client if it uses resources which have been made available by a remote system.

Like any system having a local mass storage facility, BS2000 can be both a client and a server at the same time if it is both accessing the resources of other systems and making available local resources for other systems.

Share and mount

An NFS server makes its local file hierarchy or parts thereof available for processing by NFS clients in other systems by means of the *share* command. The NFS server can share any parts of its file hierarchy. But these parts must not overlap.

An NFS client wishing to process resources made available by a remote system must mount these resources in its own file hierarchy by means of the *mount* command and is then able to access them as if they belonged to the local hierarchy. The name of the directory under which the remote resource is mounted is known as the mount point. If files and directories already exist in this directory, they are concealed by the mounted remote resource.

Transparent access

NFS clients mount the remote resources in their local file systems. On mounting, no copy of the remote resource is created. Processing of the remote resource is carried out through a series of procedure calls (RPCs), but these remain hidden to the user of the local system.The user is unable to perceive any difference between local and remote resources. The user needs to deal with only a single file hierarchy in which all files and directories can be processed in the same manner.

The owner and administrator of the "physical" files is the system which makes the files available for remote systems by means of the *share* command, i.e. the respective NFS server.

2.1.2 NFS protocol versions

With NFS V3.0 the NFS protocol versions 2 and 3 can be used simultaneously. When a connection is established between the client and server, the computers agree on which NFS protocol version can be used (see also the *v2* parameter of the *mount* command).

NFS V3.0 is based on the UDP protocol (User Datagram Protocol). This is a connection-less (unreliable) protocol, i. e. data packets are sent without the transport layer checking whether or not they actually reach the destination computer. Any communication problems that arise are regulated by a higher network layer, i.e. the NFS itself. If the server does not respond within a specific period, the client repeats the request and an error message is output if necessary. This ensures secure data transmission, at the cost of additional network load.

In comparison with the NFS protocol version 2, version 3 of the protocol achieves better performance without compromising reliability of network access. This is primarily due to saving on calls for determining file attributes.

NFS V3.0 also allows you to mount file systems containing files larger than 2 GB.

2.1.3 Read and write process with NFS

The write process

The speed at which an application can perform write processes is measured by the time it takes to transfer the data to a secure medium, i.e. normally a hard disk.

NFS protocol version 2 used synchronous write mode. With this procedure, an application on a client writes a write process in its local NFS cache in the main memory and forwards it to the respective server. In this case, each write access of the application is converted to a write access on the NFS server. The server receives the write requests and executes them on its hard disk. It then notifies the client that it has executed the requests so that the client can remove the data from its cache and the application can conclude the write process positively. If the server does not announce the execution of requests within a certain time, the client simply re-sends the data from the cache.

NFS protocol version 3, on the other hand, uses asynchronous write mode (safe asynchronous write). As with NFS protocol version 2, write accesses are saved on the client in the cache and transmitted to the server. In this case, however, the server can send confirmation of receipt back to the client immediately without actually having to perform the write access. As far as the application on the client is concerned, this confirmation from the server indicates that the write process has been executed, and allows the application to continue working. For security reasons, the data initially remains in the NFS cache of the client. To improve efficiency, the server can collect several write processes from various clients in its cache and then write them together to the hard disk as one write process. The NFS system of the client later requests confirmation of the write process (commit). However, separate confirmation is not required for each write process. As soon as the data is on the hard disk, this request is answered positively by the server. The client or application can close a file and delete the corresponding NFS packets from the cache as soon as the respective confirmation has been received (close-to-open). If the server encounters a problem, the client can transfer the requests from its cache again. If the problem cannot be solved (e. g. if there is not enough memory available), an application error message can be output before the application terminates or before the file can be closed.

Read process

The read speed is defined as the length of time an application must wait before it is supplied with the desired data. In this case, the same procedures are used by NFS protocol version 2 and 3. Firstly, read data is buffered in the main memory of the respective client, and secondly a special read process is performed whereby more data is read into the cache than is actually needed (read ahead). This means that it may also be possible to perform the next read access of the application at the same time.

2.1.4 The Network Lock Manager

Processes that work on shared files use locks to synchronize accesses to these files. Locks on files or file areas (records) are set, released or queried with the *fcntl()* system call. The NFS protocol does not support the setting of locks because it is created as a "stateless" entity: if the NFS client triggers an action on the server, this action does not generate a state on the server that would be lost if the server crashed. For this reason, either

- no state is generated on the server (e.g. when reading data), or
- a "permanent" state is generated on the server (e.g. when creating or deleting files); permanent states continue to exist even after a server crash

From the point of view of an NFS client, a server that restarts following a crash merely appears to be a server that operates very slowly. Similarly, a client that restarts after a crash simply appears to the NFS server as a client that has not sent any requests for a long time. This convenient restart procedure makes NFS very robust and is due to the statelessness of the NFS protocol.

However, it is not possible to set locks with a stateless protocol. Locks must be registered with the NFS server and are lost if the server crashes (unless each lock is saved in non-volatile storage). Working with locks on an NFS is thus implemented in a separate "NLM" (Network Lock Manager) protocol. Only "advisory" locks are supported by the Network Lock Manager. This means that the locks are only effective when all user programs work with locks for accessing the respective files or file sections. The *lockdsrv* program implements the NLM protocol on the server side. NLM is implemented in the system kernel on the client. Nevertheless, the client still relies on the help services provided by the *lockdclnt* process running on the client.

When working with locks using NFS, it must be ensured that the client processes buffer every access to a shared file. When NFS locks are set, the NFS client does not buffer read and write data. However, if both unbuffered and buffered accesses are used (e.g. those performed before any locks were set), the consistency of shared files can no longer be guaranteed. Exception: all client processes are working on the same client computer (i.e. are also sharing the buffer).

If an NLM client or server fails, cleanup actions must be started on the respective partner when the client or server restarts. A server must delete all the locks of a restarting client; a client must again set its locks if the server restarts, in order to restore the status in place before the server crashed.

Restart actions may be required for some status-based network services. These actions are thus registered with a service independent of NLM, known as the NSM (Network Status Monitor). The NSM then ensures that the relevant actions are performed when a crashed partner restarts. The NSM protocol is implemented in the *statd* program. At present, the NLM (*lockdsrv*, *lockdclnt*) is the only service that uses the NSM (*statd*). See also the description of *lockdsrv* and *lockdclnt* in the section "Daemons".

2.1.5 The Status Monitor

The Status Monitor (implemented as *statd*) is so general that it can also support other status-based network services and applications. The recovery of lost status information following a system crash is normally one of the most difficult factors in developing network applications. Thanks to the Status Monitor, it is more or less a routine task.

The Status Monitor acts as a central collection point for network status information. It is implemented as a daemon process and uses a simple protocol that allows applications to query the status of other systems with ease. Implementing the Status Monitor renders the network less prone to errors and helps to avoid situations where applications on different systems (or even on the same system) do not agree on the status of a computer. With many applications, such situations lead to inconsistencies.

To obtain information from the Status Monitor regarding changes to the network status, all systems that are to be monitored by the Status Monitor must be registered with the monitor by the application. If any of these systems fails (or, more precisely, if any of these systems restarts following a crash), all applications that have registered this system with the Status Monitor are informed about the restart by the monitor. The applications can then employ appropriate measures to update their status information.

This approach offers the following advantages:

- The time and code outlay resulting from cooperation with the Status Monitor must only be borne by applications that use status-based network services.
- Implementation of status-based network applications is simplified, because the Status Monitor shields the application developer from the complexity of the network.

For more information on the Status Monitor, see the description of *statd* in the section "Daemons".

2.1.6 Files larger than 2 Gbytes

NFS V3.0 supports files that are bigger than 2 Gbytes (large files) and file systems that are bigger than 2 Gbytes (large file systems), i.e. file sizes and file offsets are 64-bit values.

It is possible to set up large file systems locally and make them available to NFS clients.

Support for large files covers all accesses where file areas are addressed: reading, writing, querying and changing the file size, as well as setting file locks.

Accesses to NFS files are always initiated by the NFS client and performed by the NFS server. If both the client and server support large files, large files can be processed without restriction. However, problems arise if either the client or the server does not support large files.

Below is a description of the behavior shown when different operating system versions and NFS versions are implemented. The behavior described here may deviate from the behavior of systems with different implementations (e.g. systems from other manufacturers).

32-bit clients and 64-bit servers

If a 64-bit server is serving 32-bit clients, the client may not be able to interpret certain file or file system parameters, e.g. the size of a file. System calls from the client may fail in such cases.

Exception:
All clients for which a large file system is provided are normally permitted to mount this, regardless of whether the file system parameters can be displayed on the clients or not.

Behavior with respect to clients with NFS protocol version 2

If POSIX provides a large file system for NFS clients, protocol version 2 clients are also permitted to mount this file system. The file system parameters (number of blocks, etc.) are displayed correctly up to a file system size of 1 Tbyte. However, the server will reject all client accesses to large files, because the client cannot display the file size in 32-bit representation.

Behavior with respect to clients with NFS protocol version 3

If NFS protocol version 3 is used, the server cannot distinguish between 32-bit and 64-bit clients. It is therefore up to the 32-bit client to decide how to behave in relation to file or file system parameters that cannot be displayed in 32-bit representation. Normally, the client will allow a system call to fail if the result parameters cannot be expressed in 32-bit notation.

2.2 NFS in BS2000

NFS in BS2000 belongs to the POSIX program packages. The prerequisites for the use of NFS in BS2000 and for the inter-operation of NFS and POSIX are described in the following.

2.2.1 Network connection

Refer also to the "BCAM User Guide".

A network with a TCP/IP communications capability is prerequisite for working with NFS. For this you will need the openNet Server product. BCAM is required to connect a BS2000 computer to a TCP/IP network. BCAM is a component of openNet Server.

Any other computers in the network which similarly have a TCP/IP interface and on which NFS or a compatible software product for distributed file access is installed can be NFS partners in BS2000.

2.2.2 POSIX

Refer here also to the manual "POSIX Basics".

In BS2000, POSIX (**P**ortable **O**pen **S**ystem **I**nterface for UNI**X**) offers an environment similar to UNIX which complies with the POSIX Standard and the XPG4 Standard. POSIX in BS2000 is essentially delivered in two products:

- POSIX-BC contains the POSIX shell with a basic complement of POSIX commands (also called the basic shell) and the POSIX subsystem
- POSIX-SH contains additional POSIX commands

POSIX-BC is prerequisite for NFS. The components of POSIX-BC described in the following are used by NFS or when working with NFS.

POSIX shell

The shell program in BS2000 provides a command interface similar to UNIX in which POSIX commands can be entered and POSIX programs can be started.
The shell is started in BS2000 command mode with: `/START-POSIX-SHELL`
The shell is terminated with the POSIX command: `exit`

POSIX subsystem

The POSIX subsystem is a TPR subsystem which processes the requests of privileged and non-privileged users. It performs the tasks of the UNIX system kernel in BS2000. It consist of:

- a UNIX system kernel which has been ported into BS2000
- BS2000 interfaces and services which establish a connection between the ported UNIX system kernel and BS2000
- routines for initializing and terminating the POSIX subsystem

C library functions

CRTE also provides the C library functions of the POSIX program interface in addition to the BS2000 C library functions. This enables programs in BS2000 to implement the functionality required by the POSIX Standard and, for example, to process POSIX file systems.

POSIX file system

With POSIX, another file system is available in BS2000: the POSIX file system. This corresponds to a UNIX file system. NFS works with this file system.

A POSIX file system is created by the POSIX installation program. It is stored in a BS2000 PAM file (Primary Access Method). Distribution of the file hierarchy over a number of PAM files in BS2000 corresponds to the distribution in partitions in UNIX. PAM files in which a POSIX file system is situated are also referred to as container files.

The POSIX file system comprises directories and files. The files of the POSIX file system are byte-oriented. POSIX handles files as standard in EBCDIC format.

2.2.3 Components of NFS

NFS consists of commands and daemons and also uses certain administration files:

Administration files	Commands	Daemons
/etc/mnttab /etc/rmtab /etc/vfstab /etc/dfs/dfstab /etc/dfs/fstypes /etc/dfs/sharetab	dfsmounts dfsshares mount, mountall nfsstat share, shareall showmount umount, umountall unshare, unshareall	biod mountd nfsd pcnfsd statd lockdsrv lockdclnt

Commands

NFS offers commands which make available (share) local resources, mount remote resources, or output information about shared or mounted resources.

The commands *mount*, *mountall* and *umount*, *umountall* are contained in their basic form in POSIX-BC. With NFS, these commands can be used with additional options and/or with NFS-specific functionality.

Daemons

Daemons are system processes which run permanently and normally in the background, and perform universal tasks. The NFS daemons coordinate the operations taking place over the network, such as mounting and the input /output actions. The NFS daemons are started automatically when NFS starts.

Administration files

The administration files support the management of resources. They contain either information for the user which is output by means of commands, or information for commands which either the user or commands entered in these files.

RPC (remote procedure call)

NFS uses remote procedure calls (RPCs) for network communication. To this end, the *rpcbind* daemon and the *rpcinfo* program are supplied with POSIX-BC. The *rpcbind* daemon determines the addresses of the communications partners. The *rpcinfo* program gives information about the connections, transport routes, programs etc. which are used in the network communication.

2.2.4 Interaction of NFS and POSIX

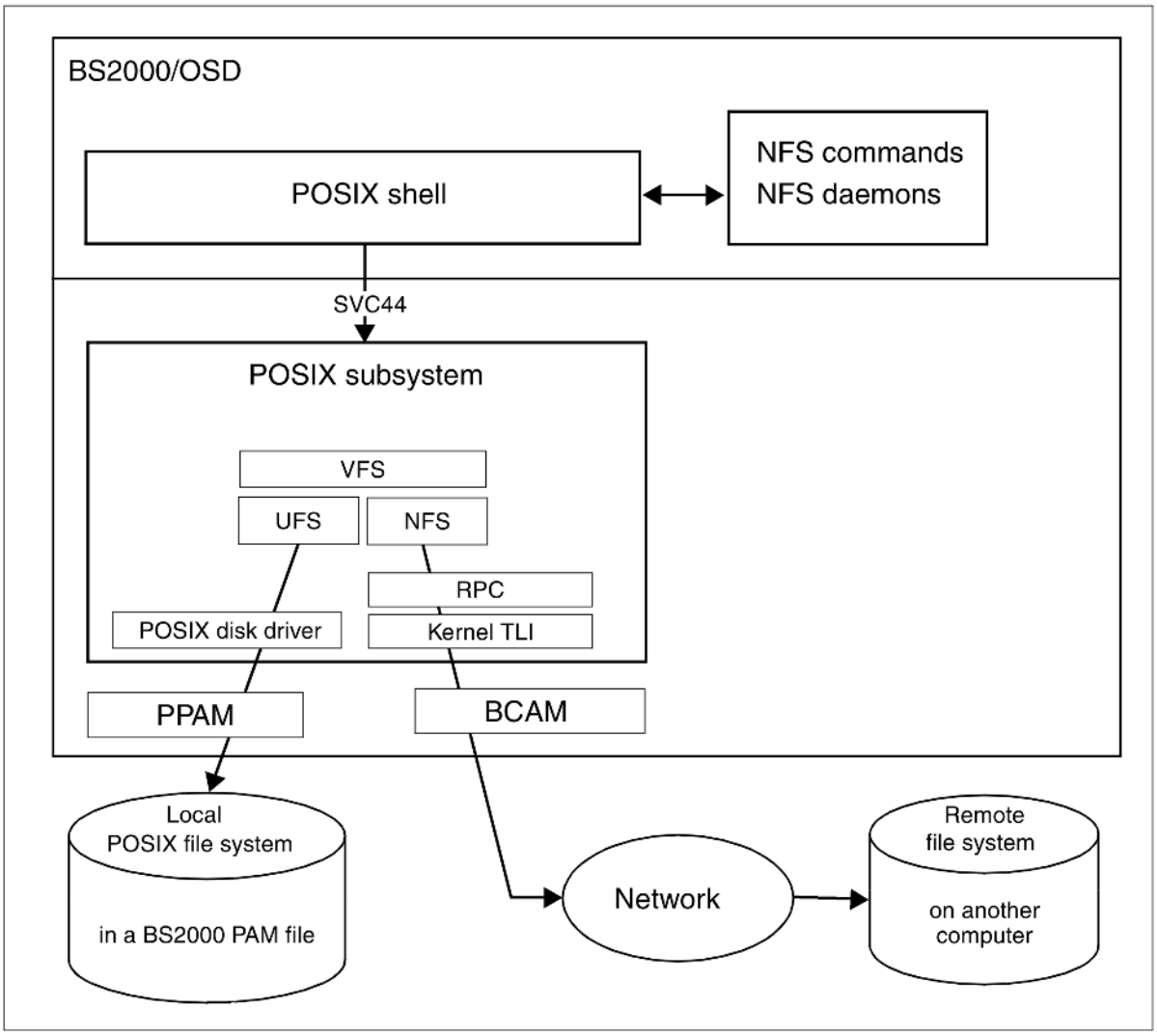

Figure 3: NFS and POSIX (view as client)

As a POSIX program package, NFS depends on POSIX and utilizes the following POSIX functionality:

- The NFS commands are entered in the POSIX shell.
- Among other things, remote file system requests are processed in the POSIX subsystem.

For example, a file system request is first passed to the POSIX subsystem component VFS (Virtual File System). On the basis of the file system type, this recognizes whether a local or remote file system is to be processed:

- If a local file system is to be processed, the request is dealt with by way of the POSIX component UFS (UNIX File System), the POSIX disk driver and the BS2000 component PPAM (Privileged Primary Access Method).
- When a remote resource is to be processed, the request is handled through NFS. Communication with the remote computer is via RPCs (Remote Procedure Calls) and Kernel TLI (Transport Layer Interface) and the BS2000 component BCAM.

2.3 Security

Using NFS, from computers in a local network it is possible to access files belonging to a remote computer; shared use can be made of the data contained in the network. Unauthorized access can be prevented by using the various protection mechanisms of BS2000, POSIX and NFS. The following protection mechanisms are available:

- user administration (BS2000)
- file access protection (BS2000, POSIX and NFS)
- data backup (BS2000)
- port monitoring (NFS)

2.3.1 User administration

Refer also to the manual "POSIX Basics".

The BS2000 user administration facility SRPM (System Resources and Privileges Management) controls access to BS2000 and to POSIX. Each user must be entered in the BS2000 user catalog. The BS2000 system administrator is responsible for the administration of BS2000 users.

The POSIX user administration facility is integrated in the BS2000 user administration facility and is not, as is usual in UNIX, handled through the */etc/passwd* file. Given suitable authorization, the following SDF commands are provided for POSIX user administration; these can be used, for example, to assign the POSIX user number (uid):

```
/ADD-USER
/MODIFY-USER-ATTRIBUTES
/SHOW-USER-ATTRIBUTES
/ADD-POSIX-USER
/MODIFY-POSIX-USER-ATTRIBUTES
/SHOW-POSIX-USER-ATTRIBUTES
/MODIFY-POSIX-USER-DEFAULTS
/SHOW-POSIX-USER-DEFAULTS
```

For POSIX group administration and administration of access from remote computers with *ssh* etc., there are other methods, commands and operands which are described in the manual mentioned above.

Root authorization (uid=0 and gid=0)

A user has root authorization when that user has the POSIX user number (uid) 0 and the POSIX group number (gid) 0 assigned. Root authorization is required for starting NFS and for the internal POSIX administration of file systems. Root authorization is assigned to the SYSROOT user ID by default. Root authorization is assigned to the TSOS user ID on installation of POSIX.

This root authorization is applicable only to the local computer. Users with root authorization from remote systems are of equal status to non-privileged users in the local system. If a user with remote root authorization is to work under user number 0, that user can be granted the appropriate authorization by means of *share* command options:

```
$ share -F nfs -o ...,root=remote_system ...
```

TSOS user ID

In addition to root authorization, the BS2000 privilege TSOS linked to the BS2000 user ID TSOS is required for all administration tasks where management of BS2000 files belonging to other user IDs must be carried out, e.g. setting up container files for storing file systems which are to be newly created.

2.3.2 File access protection

Refer also to the manual "POSIX Basics".

POSIX files and POSIX directories can be protected against unauthorized access within POSIX by means of protection bits. The container files which contain the individual file systems of the POSIX file tree can be additionally protected in BS2000 by means of appropriate file attributes. For distributed resources, access controls can additionally be implemented when such resources are made available and/or mounted.

Access protection for container files

The POSIX installation program creates container files having the attributes USER-ACCESS=*OWN and ACCESS=*WRITE for the specified user ID. These attributes should not be modified. Moreover, no file password may be assigned.

The user of a POSIX file does not require any access right for the container file in which the POSIX file is located.

Protection bits

Access protection for files and directories is implemented in POSIX by means of protection bits, as are usual in UNIX. There are three protection bits, which can be individually specified for each of the three user classes, and one identification character:

Identification character	Owner	Group	Others
`-` regular file	`r` read	`r` read	`r` read
`d` directory	`w` write	`w` write	`w` write
`l` symbolic link	`x` execute	`x` execute	`x` execute
`p` named pipe (FIFO file)			
`b` block-oriented device			
`c` character-oriented device			

Example 1

```
- rwx r-- r--
```

These characters relate to a file which the owner may read, write to and execute; other members of the group and other users may only read the file.

Example 2

```
d rwx r-- r--
```

These characters relate to a directory in which the owner can read and create/delete entries, and on which the owner can perform search operations; other members of the group and other users may only read entries in the directory.

Access protection for remote resources

The commands for making available (sharing) and mounting resources offer options which allow differential control of client access. Certain levels of authorization which can be granted to all or selected clients are listed below:

- root authorization
- read-only access
- read and write access

Through a combination of user authorizations and file access protection mechanisms it is possible to achieve the required level of protection for distributed resources in any situation.

Example

The directory */usr1/v1* belongs to the user having *uid=4712*.
It has the following protection bits: `d rwx r-- r--`
It is made available for remote NFS clients with read and write access by means of the command `share -F nfs /usr1/v1`. Any clients which mount this resource then only have read access to it. Client processes that are supposed to have write access to the directory must be logged in with the same *uid* on their computer as the owner of the directory (i.e. *4712*) because the protection bits do not permit write access by *group* and *others*.

2.3.3 Port monitoring

In communication involving TCP/IP, applications in the network are addressed through a combination of the Internet address and a port number, which uniquely identifies the recipient or the sender of a data packet. The Internet address addresses the computer, and the port number addresses the application within the computer.

An additional security feature offered by NFS is port monitoring. This is active by default. When port monitoring is active, the NFS server checks the port numbers to which an NFS client sends its request. For each client access it checks whether the port number from which the client request arrives is privileged, i.e. is less than 1024. If it is not privileged, the client request is rejected by the server.

Port monitoring can be activated and deactivated by means of the *PORTMON* parameter in the POSIX information file *SYSSSI.POSIX-BC.vvv* (refer to "Installation of NFS"):

`PORTMON=1`

: Port monitoring is activated; client requests from unprivileged port numbers are rejected; default

`PORTMON=0`

: Port monitoring is deactivated

> In a BS2000 system, privileged port number ranges can also be set differently. However, NFS always considers port numbers less than 1024 to be privileged.

3 Installation and use

This chapter describes how to install NFS, start and terminate the POSIX shell, start and terminate NFS, make available (share) and mount resources, and what you have to consider when working with POSIX files in BS2000.

3.1 Installation of NFS

The installation of NFS is an integral part of the POSIX installation program. The steps making up the installation procedure are described in the following.

Network connection

Integration of the BS2000 computer into a TCP/IP network is prerequisite for the use of NFS.

Loading the delivery unit files

The files of the NFS delivery unit must be loaded under the default user ID ($.). The installation must be carried out with the IMON installation monitor.

The NFS delivery unit comprises the following files:

`SINLIB.NFS.030`	Library containing commands, daemons, scripts etc.
`SYSSII.NFS.030`	IMON information file

Installing and starting POSIX

Refer here also to the manual "POSIX Basics".

Authorization:
The BS2000 privilege TSOS (linked to the user ID TSOS), the BS2000 privilege SUBSYSTEM-ADMINISTRATION and root authorization for the POSIX file system are required for installing and starting POSIX.

An installation program is supplied with POSIX. This program allows the following operations:

- installation of the POSIX subsystem (Install POSIX subsystem)
- administration of the POSIX file systems (Administer POSIX file systems)
- installation of the POSIX program packages (Install packages on POSIX)
- de-installation of the POSIX program packages (Delete packages from POSIX))

After initial installation of POSIX has been completed, the POSIX subsystem is active and the POSIX file systems / (root directory) and /var have been set up.

The POSIX subsystem can now be activated and deactivated as required. The following SDF commands can be used for this purpose:

`/START-SUBSYSTEM POSIX`	Activate the POSIX subsystem
`/STOP-SUBSYSTEM POSIX`	Deactivate the POSIX subsystem
`/SHOW-SUBSYSTEM POSIX`	Information about the POSIX subsystem
`/SHOW-POSIX-STATUS`	Display status of the POSIX subsystem

The initial installation of POSIX must be completed and the POSIX subsystem must be active before the POSIX program package NFS can be installed.

Modifying parameters

Parameters, such as the port monitoring facility, are modified in the information fileSYSSSI.POSIX-BC.vvv. The modified parameters take effect the next time the POSIX subsystem is started.

Installing NFS

Authorization:
Installing of NFS requires the BS2000 privilege POSIX-ADMINISTRATION and Root authorization. Both applies to the BS2000 user IDs TSOS and SYSROOT, by default.

NFS is installed using the POSIX installation program. Enter the following command:

```
/START-POSIX-INSTALLATION
```

The following main menu is then displayed:

```
                      BS2000 POSIX installation program



                               Please select




                      Administer POSIX filesystems

                      Install packages on POSIX

                      Delete packages from POSIX









Select:  MAR + ENTER                                              Exit: F2
Help  :  F1
```

Select the menu *Install packages on POSIX*.

The following sub menu then appears:

```
                      BS2000 POSIX package installation


IMON support ?     : Y  (y) mandatory for official package
                        (n) private package (SINLIB...)

name of product    : NFS
package of product :                      (optional for certain products)

version of product :          (format Vmm.n or mmn)

correction state   :          (format aso, optional for IMON support)

installation userid:          (mandatory for no IMON support)









install: ENTER    help: F1     terminate: F2
```

Enter "NFS" for the product name. The product version 030 is automatically supplemented by IMON.

Confirm the installation by pressing the ENTER key.

To uninstall NFS, use the corresponding menu *Delete packages from POSIX*.

3.2 Starting and terminating the POSIX shell

Refer here also to the manual "POSIX Basics".

The POSIX shell is required for starting NFS, for starting individual NFS daemons, and for entering NFS commands and calling the *rpcinfo* program. The POSIX shell can only becalled if POSIX has been installed and the POSIX subsystem is active.

The POSIX shell is started by means of the BS2000 command: `/START-POSIX-SHELL`

The POSIX shell is terminated with the shell command: `exit`

3.3 Using NFS

The particular tasks involved in implementing NFS depend on the requirements of your organization and the role of your system within the network.

The administration of a computer following configuration involves the following tasks:

- starting and stopping NFS
- creating and converting POSIX file systems
- providing resources and withdrawing resources
- mounting and unmounting resources
- editing administration files if the lists of resources that are to be provided automatically or are to be mounted must be updated
- providing information on released and mounted resources
- locating and eliminating errors in the operation of NFS (see "Troubleshooting, performance enhancement")

3.3.1 Starting and stopping NFS

NFS is started automatically by a start script when the POSIX subsystem is started, and is stopped by a stop script when the POSIX subsystem stops.

At startup time, all resources entered in the */etc/dfs/dfstab* file with the *-F nfs* option are made available (see "Administration files").

However, you can also start or stop NFS manually if necessary.

Starting NFS manually

Authorization: Root authorization is required for starting NFS.

Start the POSIX shell and run the start script:

```
/START-POSIX-SHELL
$ /etc/rc2.d/S20nfs start
*** starting NFS V03.0A49 ***
```

> In this case, the daemons run with the same task properties (task type, CPU limit) as the POSIX shell has. Alternatively, you can let the init process (PID 1) start NFS. Then the daemons run with the same task properties as the init process. The shell command for this type of NFS start is:
>
> ```
> $ kill -s TRAP 1
> ```

All daemons are started automatically when NFS is started. All NFS resources according to the */etc/dfs/dfstab* file are made available. Following command can be entered in order to mount NFS resources predefined in */etc/vfstab* with *automount=yes*:

```
$ mountall -F nfs
```

Terminating NFS manually

Authorization: Root authorization is required for terminating NFS.

NFS is terminated by calling the stop script:

```
$ /etc/rc0.d/K20nfs stop
*** NFS V03.0A49 going down ***
```

When NFS is terminated, the availability of resources is revoked, all file systems of type *nfs* are unmounted and all daemons are terminated.

3.3.2 Special features of the POSIX file system

Creating file systems

Refer here also to the manual "POSIX Basics".

Authorization: The BS2000 privilege TSOS and root authorization are required for the creation and administration of POSIX file systems.

File systems are created by using the POSIX installation program, menu: *Administer POSIX file systems*. After initial installation of POSIX has been completed, the POSIX file tree in its minimum configuration contains the root file system */* and the file system */var*, each in their own separate container file. Further file systems can be created as required.

When a new file system is created, the size of the container file is also defined.

Storing the file systems

The entire POSIX file tree is stored in one or several container files (these are BS2000 PAM files). The container files reside on disks (PVS, Public Volume Set).

Naming conventions

The names of POSIX files may have a maximum length of 1024 characters. This length includes the names of the directories, the file name itself, along with the delimiting slashes.

3.3.2.1 Code conversion

Refer here also to the manual "POSIX Commands", command *iconv*.

The files of the POSIX file system are normally handled by POSIX in EBCDIC format according to EDF03 or EDF04. The POSIX command *iconv* is available for conversion to ASCII format in accordance with ISO 646 or ISO 8859-1.

The code tables are contained in the directory */usr/lib/iconv*.

When resources of the POSIX file system are made available for UNIX or Windows computers, the files should be converted. Conversion is similarly required if files in ASCII format from remote computers are to be mounted in the POSIX file system.

If ASCII files are to be converted automatically, the shell variable IO_CONVERSION mustbe set to YES. Use the following command for this purpose:

```
$ IO_CONVERSION=YES
$ export IO_CONVERSION
```

Example

The content of the file *bs2000* is to be converted from ASCII to EBCDIC and the result is to be written to the file *bs2000.conv*:

```
$ iconv -f 646 -t edf04 bs2000 > bs2000.conv
```

3.3.2.2 BS2000 files

BS2000 files can be copied into the POSIX file system using the POSIX command *bs2cp* or the BS2000 command COPY-POSIX-FILE and thus be made available for other systems in the network with NFS. Refer to the manuals "POSIX Basics" and "POSIX Commands", command *bs2cp*.

BS2000 files that have been made available in POSIX using the BS2000 file system *bs2fs* can also be made available to other systems in the network using NFS. Refer to the manual "POSIX BS2000 file system bs2fs", section "Access to bs2fs file systems via NFS".

3.3.3 Sharing and unsharing resources

Authorization: Root authorization is required for making available (sharing) resources, and for revoking their availability (unsharing).

Sharing (exporting) local resources

An NFS server makes local resources available for processing by remote systems (NFS clients). You make local resources availble by means of the commands *share* or *shareall*:

- The *share* command is used to make an individual resource available for client access and to define the access rights for clients.
- The *shareall* command allows multiple resources to be simultaneously made available for client access. The command expects one or more *share* commands in a user-defined file, from the standard input, or in the administration file */etc/dfs/dfstab*.

Unsharing resources

The availability of resources is revoked either by means of the commands *unshare* or *unshareall*, or automatically on termination of NFS.

Sharing resources permanently

If you want to make certain resources available to other computers on a permanent basis, you can configure your system such that these resources are automatically shared when NFS is started. This procedure should be followed if a resource must always be available to other computers, for example, and client access will probably never or only rarely be canceled.

Resources that are to be shared automatically are entered in the */etc/dfs/dfstab* file. This file is created automatically when NFS is installed.

The *dfstab* file contains a list of all resources that are to be made available to the clients after NFS is started, and also indicates the access authorizations valid for these resources. To enter a resource in *dfstab*, delete a resource, or change options, edit the file using any supported text editor. The changes made will come into effect as soon as POSIX is restarted, a *shareall* command is issued, or NFS is stopped and restarted.

3.3.4 Mounting and unmounting resources

Authorization: Root authorization is required for mounting and unmounting resources.

Mounting resources

Without NFS it is possible to mount and unmount only local resources, while the use of NFS also allows mounting and unmounting of all remote resources made available by other systems in the network.

An NFS client mounts resources from a remote system (NFS server) in its own local file hierarchy. This presupposes that the NFS server is available and has shared the desired resource. The NFS client must have authorization to access this resource and the mount point must exist.

Should local files and directories be present in the directory which is chosen as the mount point, these will be concealed by the mounted remote resource.

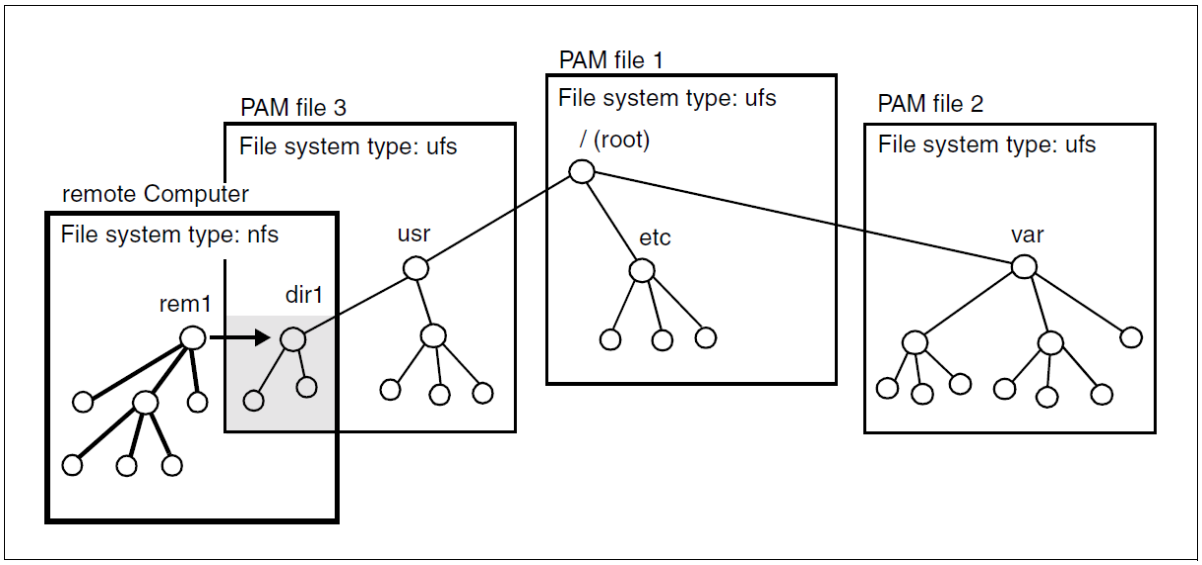

Figure 4: File systems in the POSIX file hierarchy

In figure 4 the local file hierarchy comprises three POSIX file systems and one remote file system. The remote resource *rem1* is mounted in the directory *dir1*, which is the "mount point". The two local files in *dir1* are concealed by the mounted remote resource.

Remote resources are mounted by means of the commands *mount* or *mountall:*

- The *mount* command is used to mount an individual resource in the local file system and to define the access rights for the users of that system.
- The *mountall* command allows simultaneous mounting of multiple resources. The command expects to take its specifications concerning the resources to be mounted from a user-defined file, the standard input, or the administration file */etc/vfstab*.

Unmounting resources

Remote resources are unmounted either by means of the commands *umount* or *umountall*, or automatically on termination of NFS.

Automatic mounting of remote resources

The */etc/vfstab* file can be used to determine that a file system will be mounted automatically when POSIX is started. If a file system or directory of another computer is entered in this file, the resource is mounted automatically when NFS is started. Before you can mount a remote resource automatically, you must create a mount point using the *mkdir* command and edit the */etc/vfstab* file.

3.3.5 Information about resources

The commands *share*, *mount*, *dfshares, showmount* and *dfmounts* can be used to ascertain which remote resources it is currently possible to access, which resources are mounted on the local system and have been made available for client access, and which local resources have been mounted on remote clients.

Information about shared resources

- The *share* command without operands shows all the local resources which may be accessed by clients.
- The administration file */etc/dfs/sharetab* contains information about all the local resources which have been made available by means of the *share* command.
- The *dfshares* command shows all the remote resources which have been made available for client access.

Information about mounted resources

- The *mount* command without operands shows all the local and remote resources which are mounted in the local file system. Refer to the manual "POSIX Commands", command *mount*.
- The *dfmounts* command indicates which resources from one NFS server or all NFS servers in the network are mounted on clients.
- The *showmount* command indicates on which clients resources from the local NFS server or the specified NFS server are mounted.

4 Commands, daemons and administration files

For the administration of a distributed file system, NFS provides support through:

- commands
- daemons
- the program *rpcinfo*
- administration files

4.1 NFS commands

This section describes the NFS commands in alphabetical order.

Figure 5: NFS Commands

The commands shown are those required for working with NFS. The commands are entered in the POSIX shell just like POSIX commands.

The commands *mount* and *umount* are included in the basic configuration of the POSIX commands, but only in their generic form for processing local file systems. With NFS, these commands receive additional options and functions.

The commands *mountall* and *umountall* are similarly included in the basic configuration of the POSIX commands. For NFS they require no additional functionality since they obtain their specific functionality from an input file.

The NFS commands are listed in the following overview:

Command	**Function**
`dfmounts`	Output information about mounted resources
`dfshares`	Output information about available resources
`mount`	Mount remote resources
`mountall`	Mount multiple remote resources
`nfsstat`	Output statistical information

`share`	Make local resources available for client access
`shareall`	Make multiple local resources available for client access
`showmount`	Output information about NFS clients and resources
`umount`	Unmount remote resources
`umountall`	Unmount multiple remote resources
`unshare`	Make local resources unavailable
`unshareall`	Make multiple local resources unavailable

4.1.1 dfmounts - Output information about mounted resources

The *dfmounts* command allows you to ascertain which resources of the local computer or ofanother NFS server are mounted on client systems.

Syntax

```
dfmounts[ -F nfs][ -h][ server][ ...]
```

-F nfs

Specifies that only information about resources of the file system type *nfs* is output. This option does not have to be specified because no other distributed file systems are currently supported and *nfs* is thus the only file system type which is listed in the file */etc/dfs/fstypes*.

-h

Suppresses output of the optional header.

server

The name of a computer which, as an NFS server, provides resources; only information about the resources provided by *server* is output. More than one *server* can be specified.

If no *server* is specified,information about the local resources currently mounted by clients is output.

Output

The information output by *dfmounts* consists of an optional header with column headings, followed by a list of lines containing the details about the individual resources:

Header:

```
RESOURCE   SERVER   PATHNAME   CLIENTS
```

RESOURCE

Name of the mounted resource, as required by the *mount* command on the client.

SERVER

Name of the system providing the resource.

PATHNAME

Path name of the resource provided, as required by the *share* command on the server.

CLIENTS

Names of the client systems on which the resource is currently mounted.

Files

/etc/dfs/fstypes

Table of installed utilities for distributed file systems.

Example

You wish to ascertain which resources the computer *d016ze04* is providing, and on which clients they are mounted.

```
$ dfmounts d016ze04
RESOURCE     SERVER PATHNAME                  CLIENTS

  -        d016ze04 /cpp32c                  d016ze07
  -        d016ze04 /dcm89q/dfue_test        abgqn508
  -        d016ze04 /dcm89q/fs               abgse404,abgse218,sqnet203
  -        d016ze04 /javaqa                  d016ze07
  -        d016ze04 /tjinst                  d016ze07
```

4.1.2 dfshares - Output information about available resources

The *dfshares* command allows you to ascertain which resources are made available on remote systems for client access.

Syntax

```
dfshares[ -F nfs][ -h][ server][ ...]
```

-F nfs

Specifies that only information about resources of the file system type *nfs* is output. This option does not have to be specified because no other distributed file systems are currently supported and *nfs* is thus the only file system type which is listed in the file */etc/dfs/fstypes*.

-h

Suppresses output of the optional header.

server

The name of a computer which, as an NFS server, provides resources; only information about the resources provided by *server* is output. More than one *server* can be specified.

If no *server* is specified, information about the resources provided by the local computer for client access is output.

Output

The information output by *dfshares* consists of an optional header with column headings, followed by a list of lines containing the details about the individual file systems.

Header:

```
RESOURCE   SERVER   ACCESS   TRANSPORT
```

RESOURCE

Name of the mounted resource, as required by the *mount* command on the client.

SERVER

Name of the system providing the resource.

ACCESS

Access authorizations for the client systems; because *dfshares* cannot determine this information for an NFS resource, a hyphen (-) is output.

TRANSPORT

Transport provider over which the file system is shared; because *dfshares* cannot determine this information for an NFS file system, a hyphen (-) is output.

Files

/etc/dfs/fstypes

Table of installed utilities for distributed file systems.

Example

You wish to ascertain which resources are available on the computer *d016ze07*. The information is to be output with a header.

```
$ dfshares -F nfs d016ze07
RESOURCE                                  SERVER ACCESS    TRANSPORT
  d016ze07:/posix315                    d016ze07  -         -
  d016ze07:/vgt                         d016ze07  -         -
  d016ze07:/sm2ibis                     d016ze07  -         -
  d016ze07:/ccsxtest                    d016ze07  -         -
```

4.1.3 mount - Mount remote resources

The *mount* command allows mounting of both local and remote resources. Before remote resources can be mounted, the remote NFS server must first make available the desired resource for client access and that resource must be accessible. The remote resource is mounted in the local file hierarchy at the path name position *mountpoint* . The directory *mountpoint* must already exist. If *mountpoint* already contains data prior to the *mount* operation, this is concealed until the *resource* is unmounted again.

Only the mounting of remote resources of the file system type *nfs* is described here. For a description of mounting local resources of the file system types *ufs* or *bs2fs*, refer to the "POSIX Commands" manual, command *mount*.

If the resource is listed in the file */etc/vfstab*, it is sufficient to specify either *resource* or *mountpoint*. The command then searches in the file */etc/vfstab* for further specifications.

The *mount* command enters added file systems in the table of mounted file systems */etc/mnttab*.

If the command is invoked without options, all local and remote resources currently mounted on your system are listed.

Authorization: The mounting of remote resources requires root authorization. To enter the command without options requires no special authorization.

Syntax

```
mount[ -F nfs][ -r][ -o specific_options][ resource[ mountpoint]]
```

-F nfs

Specifies that a resource of the file system type *nfs* is to be mounted. If this option is not specified but *resource* or *mountpoint* is specified, the command searches in the file */etc/vfstab* for a corresponding entry and mounts the resource with the file system type specified there.

-r

Mounting the resource with read authorization.

-o specific_options

A list of file system specific options which can be specified after *-o*. The individual options in the list are separated by commas, and are described below.

resource

Specifies the resource to be mounted. Remote resources are specified in the form *server:pathname* where *server* is the computer name of the NFS server providing the resource and *pathname* is the absolute path name of the resource.

mountpoint

Specifies where the *resource* is to be mounted locally. An absolute path name must be specified.

The following *specific_options* can be specified after *-o*:

rw | **ro**

rw defines read and write access, *ro* defines read-only access to the mounted *resource*. The default is *rw*.

suid | **nosuid**

Specifies whether set s-bits are to be taken into consideration during execution (*suid*) or ignored (*nosuid*). The default is *suid*.

remount

Effects remounting of an already mounted resource if only access authorizations have been changed.

bg | **fg**

Specifies whether, if the first mount attempt fails, a new attempt is to be made in the background (*bg*) or in the foreground (*fg*). The default is *fg*.

retry=*n*

Specifies how often an unsuccessful mount operation is to be repeated.The default is 10 000.

port=*n*

Specifies the port number of the NFS server. The default is *NFS_PORT* (2049).

grpid=*GID*

Creation of a file whose group number (*GID*) corresponds to the effective *GID* of the caller. This setting can be invalidated in each directory by setting the s-bit for the group of the parent directory; in this case, the group number corresponds to the number of the parent directory. Files which are created in file systems, which are not mounted using the *grpid* option, are subject to BSD semantics, which means that the *GID* must be adopted by that of the parent directory.

rsize=*n*

Defines the size of the read buffer as *n* bytes. The default is 8 Kbytes.

wsize=*n*

Defines the size of the write buffer as *n* bytes. The default is 8 Kbytes.

timeo=*n*

Defines the maximum value for the time which the client is to wait for execution of an NFS job. *n* is specified as tenths of a second. The default is 11 tenths of a second.

retrans=*n*

Sets the number of repeated transfers for an NFS job to *n*. The default is 5.

soft | **hard**

Specifies whether, if the server does not respond, an error is to be returned (*soft*) or whether a mount operation is to be retried until the server responds (*hard*).

intr

Specifies that NFS jobs can be aborted via the keyboard. If this option is not specified, in the case of a resource mounted with the option *hard* the terminal remains blocked until the job has been processed.

noac

Suppresses the buffering of attributes in the attribute cache.

v2

Mounts resources of servers using NFS protocol version 2, even if the servers support protocol version 3.

Mounting in background or foreground

If NFS file systems are mounted with the option *bg*, this means that *mount* is to repeat the mount operation in the background if the *mountd* daemon of the server does not respond. *mount* repeats the request as often as specified by the option *retry=n*. As soon as the file system has been mounted, all NFS requests to the kernel wait *timeo=n* tenths of a second for a response. If no response is received, the wait time is multiplied by 2 and the request is transferred again. Once the number of retries reaches the number specified in the option *retrans=n*, a file system mounted with the option *soft* returns an error for the request; if the file system was mounted with the option *hard*, it issues a warning and continues to repeat the request.

Attribute cache

The attribute cache provides intermediate storage for file attributes for the client. Attributes relating to a file are deleted after a certain period of time. If a file is changed before the attribute cache is emptied, the emptying interval is extended by the time which has elapsed since the last change; this presupposes that recently changed files will soon be changed again. For normal files and for directories, minimum and maximum values apply to the extension of the emptying intervals.

Files

/etc/mnttab

Table of mounted file systems

/etc/dfs/fstypes

Table of installed utilities for distributed file systems

/etc/vfstab

Table of defined file systems

Examples

Example 1:

You wish to mount the directory */usr/src* from the remote computer *serv* on your local computer in the directory */usr1/proj3/src*. On the computer *serv* the directory has been made available by means of NFS.

```
$ mount -F nfs serv:/usr/src /usr1/proj3/src
```

Example 2:

On your computer, you wish to mount the directory */usr/man* which the computer *docgroup* makes available via NFS. The mount operation is to be retried until the server responds. You wish to mount the directory for read-only access; interruption via the keyboard is to be possible. You create the directory */home1/usr/man* as the mount point.

```
$ mkdir /home1/usr/man
$ mount -F nfs -o ro,hard,intr docgroup:/usr/man /home1/usr/man
```

4.1.4 mountall - Mount multiple remote resources

The *mountall* command allows simultaneous mounting of multiple resources. The command takes the specifications concerning the resources to be mounted from an input file, from the file */etc/vfstab* or from the standard input. Specifications from an input file or from the standard input must have the same format as the file */etc /vfstab*.

If no option is specified, all file systems which are described in the file */etc/vfstab* and for which the field *automnt* is set to *yes* are mounted.

Authorization: The command can only be entered with root authorization.

Syntax

```
mountall[ -F nfs][ -| file]
```

-F nfs

Specifies that resources of the file system type *nfs* are to be mounted.

If this option is not specified, all resources specified in the input file or via the standard input are mounted.

-

Specifies that the command expects information about the file systems to be mounted from standard input.

The individual lines are terminated with *RETURN* or with *EM DUE*.

The command is executed after entering *Ctrl-D* or *@@D*.

file

Input file containing specifications for the file systems to be mounted.

If neither - nor *file* is specified, the file */etc/vfstab* is taken as the input file by default.

Files

/etc/mnttab

Table of mounted file systems

/etc/vfstab

Table of defined file systems

4.1.5 nfsstat - Output statistical information

The *nfsstat* command outputs statistical information about the NFS communication between client and server. Information about the following is output:

- the number of RPCs (remote procedure calls) sent and received, and any errors which occurred during them. A distinction is made between those RPCs which relate to the computer as a client and those which relate to the computer as server.
- the number and type of NFS calls, and any errors which occurred during them. Here too, a distinction is made between those which relate to the computer as a client and those which relate to the computer as server.

You can also use this command to reset the statistics counters and thus define the period for the statistics.

If you enter the command without any options, all statistical information is output. The statistics counters are not reset.

Syntax

```
nfsstat[ -cnrsz]
```

-c

Outputs only information which relates to the computer as a client.

-n

Outputs only information about NFS calls.

-r

Outputs only information about RPCs.

-s

Outputs only information which relates to the computer as server.

-z

Causes the output and subsequent reset of the statistics counters to 0.

The operands listed above determine which statistics counters are to be reset to 0.

If only -z is specified, all statistics counters are reset to 0.

RPC statistics: Server

calls

Number of RPCs received

badcalls

Number of RPCs received with errors (the sum of *badlen* and *xdrcall*)

nullrecv

Number of RPCs which were unavailable although thought to have been received

badlen

Number of RPCs received with too short a length

xdrcall

Number of RPCs received whose header could not be XDR decoded

RPC statistics: Client

calls

Number of RPCs which were made

badcalls

Number of RPCs which were rejected

retrans

Number of RPC packets which had to be re-transmitted during a call because no acknowledgment or an incorrect acknowledgment was received

badxid

Number of acknowledgments for RPC packets that arrived after the RPC was already completed

timeout

Number of RPC packets transferred during a call for which no reply was received within a certain period of time

wait

Number of calls where it was necessary to wait for internal file structures

newcred

Number of RPCs for which the authentication parameters had to be refreshed by the server computer

NFS statistics

The NFS statistics are structured similarly for server and client. Information is output on the NFS calls made (*calls*), the failed NFS calls (*badcalls*), and a breakdown of the types of the NFS calls made in the form of an absolute number and a percentage.

Additionally, for the client, statistics are included on the number of requests for internal data structures (*nclget*) and the resulting wait situations (*nclsleep*).

calls

: Number of NFS requests sent

badcalls

: Number of failed NFS requests

nclget

: Details of how often internal data structures were requested

nclsleep

: Details of how often it was necessary to wait with *nclget*

NFS protocol versions 2 and 3:

null

: No action (for test purposes)

getattr

: Request attributes for a file

setattr

: Set attributes for a file

lookup

: Locate a file

readlink

: Read a symbolic reference

read

: Read in a file

wrcache

: Write to the buffer

write

: Write to a file

create

: Create a file

remove

: Delete a file

rename

: Rename a file

link

: Set simple reference

symlink

: Set symbolic reference

mkdir

: Create directory

rmdir

: Delete directory

readdir

: Read in a directory

fsstat

: Fetch file system information

NFS protocol version 3 only:

access

: Check file access rights

commit

: Stabilize write request

fsinfo

: Fetch static file system information

fsstat

: Fetch dynamic file system information

mknod

: Create special file

pathconf

Fetch information on the path configuration of a file (maximum path length, maximum number of links, etc.)

readdirplus

Extended reading of directory in accordance with NFS V3

Examples

Example 1:

You want to have all the NFS and RPC statistical information output which concerns your local computer as an NFS client.

```
$ nfsstat -c

Client rpc:
calls       badcalls    retrans     badxid      timeout     wait   newcred
510690      0           1559        338         1559        0      0

Client nfs version 2:
calls       badcalls    nclget      nclsleep
26          0           26          0
null        getattr     setattr     root        lookup      readlink
0  0%       12 46%      2  7%       0  0%       8 30%       0  0%
read        wrcache     write       create      remove      rename
0  0%       0  0%       1  3%       1  3%       0  0%       0  0%
link        symlink     mkdir       rmdir       readdir     fsstat
0  0%       0  0%       0  0%       0  0%       1  3%       1  3%

Client nfs version 3:
calls       badcalls    nclget      nclsleep
510661      0           510654      0
null        getattr     setattr     lookup      access      readlink
0  0%       152  0%     1  0%       285479 55%  2  0%       0  0%
read        write       create      mkdir       symlink     mknod
59776 11%   60164 11%   7  0%       0  0%       0  0%       0  0%
remove      rmdir       rename      link        readdir     readdirplus
66635 13%   0  0%       0  0%       0  0%       36901  7%   0  0%
fsstat      fsinfo      pathconf    commit
3  0%       3  0%       2  0%       1537  0%
```

Example 2:

You want to output all the static information about NFS and RPC in relation to your local computer as NFS server.

```
$ nfsstat -s

Server rpc:
calls       badcalls    nullrecv    badlen      xdrcall
137514      31720       0           0           0

Server nfs version 2:
calls       badcalls
1631        0
null        getattr     setattr     root        lookup      readlink
1  0%       49  3%      0  0%       0  0%       63  3%      3  0%
read        wrcache     write       create      remove      rename
4  0%       0  0%       1479 90%    10  0%      3  0%       0  0%
link        symlink     mkdir       rmdir       readdir     fsstat
0  0%       0  0%       0  0%       0  0%       17  1%      2  0%

Server nfs version 3:
calls       badcalls
104163      0
null        getattr     setattr     lookup      access      readlink
1  0%       287  0%     0  0%       102264 98%  0  0%       2  0%
read        write       create      mkdir       symlink     mknod
2  0%       1011  0%    311  0%     2  0%       0  0%       0  0%
remove      rmdir       rename      link        readdir     readdirplus
0  0%       0  0%       0  0%       0  0%       278  0%     0  0%
fsstat      fsinfo      pathconf    commit
2  0%       2  0%       1  0%       0  0%
```

4.1.6 share - Make local resources available for client access

The *share* command makes local resources available for remote client access.

If the command is entered without options, all resources which are currently made available by your system are listed.

Syntax

```
share[ -F nfs][ -o specific_options][ -d description][ pathname]
```

-F nfs

Specifies that a resource of the file system type *nfs* is to be made available. Because no other file system types for distributed file usage are supported in POSIX, this option may be omitted.

-o specific_options

A list of file system specific options which can be specified after *-o*. The individual options in the list are separated by commas, and are described below.

-d description

A text enclosed in quotes which can be used to furnish clients with comments concerning usage of the resource. The text may not contain special characters and must not exceed 32 characters in length.

pathname

Path name of the resource to be made available.

The following *specific_options* can be specified after *-o*:

rw

Makes the resource available for read and write access. This option is the default, i.e. if no *specific_options* are specified, read/write access is granted to all clients.

ro

Makes the resource available for read-only access.

rw=*client*[:*client*]...

Makes the resource available for read and write access to the listed clients. This specification overrides the sub option *ro* for individual clients.

ro=*client*[:*client*]...

Makes the resource available for read-only access to the listed clients. This specification overrides the sub option *rw* for individual clients.

anon=*uid*

NFS client processes with the user ID 0 (root) will access the shared resource with the effective user ID *uid* . The default value is `UID_NOBODY` (60001). Exceptions do apply to those client computers listet in the "*root=...*" option.

If *uid* is set to -1, access to this resource is denied.

root=*host*[:*host*]...

Specifies that NFS client processes with user ID 0 (root) from the specified hosts are also allowed to access the shared resource with root privileges. By default, no host is granted root privileges.

When sharing *bs2fs* resources, the following additional *specific_options* can be specified after *-o*:

bs2anon=*bs2000_uid*
bs2conv
bs2nameconv

For detailed information, refer to the manual "POSIX BS2000 file system bs2fs".

The command will not be executed if conflicting access rights are defined for a client. This is the case, for example, if the same client name is specified both after *ro=* and also after *rw=*, or if the options *ro* and *rw* are specified together and without arguments.

Files

/etc/dfs/fstypes

Table of installed utilities for distributed file systems

/etc/dfs/sharetab

Table of shared resources

Examples

Example 1:

You want to make available the directory */usr/reports/mtgmemos* for client access via NFS. The directory is to be made available for all clients with read-only access.

```
$ share -F nfs -o ro -d "MEMOS on project X" /usr/reports/mtgmemos
```

Example 2:

You want to make available the directory */export/graphics* for client access via NFS. All clients should have read-only access, only the client "art.dept" gets read and write permissions.

```
$ share -F nfs -o ro,rw=art.dept /export/graphics
```

4.1.7 shareall - Make multiple local resources available for client access

The *shareall* command allows multiple NFS resources to be simultaneously made available for client access.

The command takes the specifications concerning the resources to be made available from an input file, from the file */etc/dfs/dfstab* or from the standard input. Specifications from an input file or from the standard input must have the same format as the file */etc/dfs/dfstab*. The format of these entries corresponds to the syntax of the *share* command.

If the *shareall* command is entered without options, all resources which are entered in the file */etc/dfs/dfstab* are made available.

Syntax

```
shareall[ -F nfs][ -| file]
```

-F nfs

Specifies that a resource of the file system type *nfs* is to be made available. Because no other file system types for distributed file usage are supported in POSIX, this option may be omitted.

-

Specifies that the command expects that the specifications for the file systems to be made available will come from the standard input. The individual lines are terminated with *RETURN* or with *EM DUE*. The command is executed after entering *Ctrl-D* or *@@D*.

`file`

Input file containing specifications for the file systems to be made available. If neither - nor *file* is specified, the file */etc/dfs/dfstab* is taken as the input file by default.

Files

/etc/dfs/dfstab

Table of resources to be shared

Examples

Example 1:

You create an input file called *sh_list*, via which three resources are to be made available for client access. The file contains the following entries:

```
$ cat sh_list
share -F nfs -o ro /usr/reports/mtg.notes
share -F nfs -o ro,rw=art.dept /export/graphics
share -F nfs /usr/man
```

The following command makes available all resources:

```
$ shareall sh_list
```

Example 2:

You wish to make three resources available for client access. Because this is a one-off situation, no input file is required.

```
$ shareall -
```

The cursor jumps to the next line. Enter the following commands, and terminate each command line with *RETURN* or *EM DUE*.

```
share -F nfs -o ro /usr/reports/mtg.notes
share -F nfs -o ro,rw=art.dept /export/graphics
share -F nfs /usr/man
```

Press *Ctrl-D* resp. *@@D*. The commands are executed.

4.1.8 showmount - Output information about NFS clients and resources

The *showmount* command provides information on the names of those client computers which have mounted resources from the local or the specified computer. The information which is displayed by *showmount* is managed by the *mountd* daemon on the server computer and stored in the file */etc/rmtab*.

If no options are specified, the names of those client computers are output which have mounted file systems that are made available by the local computer.

Syntax

```
showmount[ -a][ -d][ -e][ hostname]
```

-a

Outputs a list in the format: *hostname:directory* where *hostname* is the name of the client computer and *directory* is the name of the mounted resource.

-d

Outputs the names of all directories of the computer *hostname* which are mounted on client computers. The names of the client computers are not included in the output.

-e

The names of the directories which are made available by the computer *hostname* are output.

`hostname`

Name of the computer about which the information is to be output.

If *hostname* is not specified, the information is output for the local computer.

If the session on a client computer which has mounted file systems from a server computer does not terminate normally, then the entry in the file */etc/rmtab* is retained until the system is restarted and the command is *umount -a* executed.

Files

/etc/rmtab

Table of mounted remote resources

Examples

Example 1:

Your local computer is called *london*. You wish to ascertain which client computers have mounted resources from your computer.

```
$ showmount
mountainview
tokyo
```

The client computers *mountainview* and *tokyo* have mounted resources made available by the computer *london*.

Example 2:

You would like to know which resources from your local computer *london* are mounted onclients.

```
$ showmount -d
/usr1/steven
```

Only the directory */usr1/steven*, which is made available by your computer, is mounted on clients.

Example 3:

You would like to know which client computers have mounted which resources from your local computer *london*.

```
$ showmount -a
mountainview:/usr1/steven
tokyo:/usr1/steven
```

The directory */usr1/steven*, which is made available by your computer, is mounted both on the client computer *mountainview* and also on the client computer *tokyo*.

Example 4:

You would like to know which directories from your local computer *london* have been made available for which client computers.

```
$ showmount -e
export list for london:
/usr1                       (everyone)
/usr                        windsor,stirling
```

The local computer *london* makes available the directory */usr1* for all clients (user group *everyone)* and the directory */usr* for the client computers *windsor* and *stirling*.

4.1.9 umount - Unmount remote resources

The *umount* command allows local and remote resources to be unmounted. It makes no difference here whether the resource was mounted explicitly with the *mount* command or implicitly via the file */etc/vfstab*.

Only the unmounting of remote resources is described here (file system type *nfs*). For a description of how to unmount local resources (file system type *ufs* or *bs2fs*), refer to the "POSIX Commands" manual, *umount* command.

Authorization: The command can only be entered with root authorization.

Syntax

```
umount[ -F nfs] resource | mountpoint
```

-F nfs

Specifies that a resource of the file system type *nfs* is to be unmounted. If this option is not specified, the file system type is determined using the file */etc/mnttab*.

`resource`

Specifies the resource which is to be unmounted. Remote resources of the type *nfs* are specified in the form *server:pathname* where *server* is the computer name of the NFS server providing the resource and *pathname* is the absolute path name of the resource.

`mountpoint`

Specifies the mount point at which the resource is to be unmounted. An absolute path name must be specified.

Files

/etc/mnttab

Table of mounted file systems

/etc/dfs/fstypes

Table of installed utilities for distributed file systems

/etc/vfstab

Table of defined file systems

4.1.10 umountall - Unmount multiple remote resources

The *umountall* command allows all resources currently mounted on your system to be unmounted.

Only the unmounting of remote resources of the file system type *nfs* is described here. For a description on how to unmount local resources of the file system types *ufs or bs2fs,* refer to the "POSIX Commands" manual, *umountall* command.

If the command is entered without options, all resources mounted on your system will be unmounted.

Authorization: The command can only be entered with root authorization.

Syntax

```
umountall[ -F nfs][ -k]
```

-F nfs

Specifies that only resources of the file system type *nfs* are to be unmounted.

-k

Sends the signal SIGKILL to all processes which have opened files in the resource to be unmounted.

Files

/etc/mnttab

Table of mounted file systems

/etc/vfstab

Table of defined file systems

4.1.11 unshare - Make resources unavailable

The *unshare* command revokes the availability of resources. This locks local resources against mount attempts by remote systems.

There must be an appropriate entry for the resource in the file */etc/dfs/sharetab*.

Syntax

```
unshare[ -F nfs] pathname
```

-F nfs

Specifies that availability is to be revoked for a resource of the file system type *nfs*. If this option is not specified, the file system type will be taken from the first line of the file */etc/dfs/fstypes*.

`pathname`

Specifies the path name of the resource which is to be locked to client access.

Files

/etc/dfs/fstypes

Table of installed utilities for distributed file systems

/etc/dfs/sharetab

Table of shared resources

Example

The directory */export/templates* which was automatically made available via *dfstab* is to be made temporarily unavailable. NFS clients are no longer to be allowed access to it.

```
$ unshare -F nfs /export/templates
```

You can make the directory available once again by means of the *share* command.

4.1.12 unshareall - Make multiple resources unavailable

The *unshareall* command simultaneously revokes the availability of multiple resources at the same time. This locks local resources against mount attempts by remote systems.

There must be an appropriate entry for the resources in the file */etc/dfs/sharetab*.

Syntax

```
unshareall[ -F fstype[,fstype...]]
```

-F fstype[fstype...]

Specifies that availability is to be revoked only for resources of the specified file system types. If this option is not specified, availability will be revoked for all resources made available by the local system.

This option currently has no meaning because *nfs* is the only type of distributed file system supported by POSIX.

Files

/etc/dfs/dfstab

Table of resources to be shared

Examples

Example 1:

You wish to make unavailable all the NFS resources made available by your system, and lock them against access by remote NFS clients.

```
$ unshareall -F nfs
```

Example 2:

You wish to make unavailable all resources currently made available by your system.

```
$ unshareall
```

4.2 Daemons

This section describes the daemons in alphabetical order.

```
/
`-- usr
    |-- lib
    |   `-- nfs
    |       |-- biod
    |       |-- lockdclnt
    |       |-- lockdsrv
    |       |-- mountd
    |       |-- nfsd
    |       |-- pcnfsd
    |       `-- statd
    `-- sbin
        `-- rpcbind
```

Figure 6: Daemons

The NFS daemons *biod*, *mountd*, *nfsd*, *pcnfsd, lockdclnt, lockdsrv* and *statd* are started automatically at NFS startup. The RPC daemon *rpcbind* is started when POSIX is started. As a general rule, the daemons should only be started automatically. However, if daemons are started individually, then a daemon is called in the POSIX shell with the appropriate path name. The daemons run as POSIX background processes partly in TU, partly in TPR. Once started, they "sleep" until they are "woken" by a request.

The daemons perform the NFS-specific functions or handle communication via RPC (remote procedure call). The daemons *nfsd* and *mountd* are responsible for the functions of the NFS server, while the *biod* daemons are responsible for the functions of the NFS client.The *rpcbind* daemon is not an NFS daemon but it is required to handle the network communication via RPC and is supplied with POSIX-BC in BS2000. For further information on the functions of the daemons, please refer to the publication "Managing NFS and NIS".

The following daemons run by default on a BS2000 system:

```
UID        PID CMD
SYSROOT     43 [rpcbind]
SYSROOT     87 [biod]
SYSROOT     86 [biod]
SYSROOT     78 [nfsd]
SYSROOT     80 [nfsd]
SYSROOT     82 [nfsd]
SYSROOT     84 [nfsd]
SYSROOT     88 [biod]
SYSROOT     85 [biod]
SYSROOT     90 [mountd]
SYSROOT     92 [pcnfsd]
SYSROOT     94 [statd]
SYSROOT     96 [lockdsrv]    (on NLM server and NLM client)
SYSROOT     98 [lockdclnt]   (on NLM client)
```

The following overview lists the individual daemons:

Daemon	**Function**
biod	NFS client daemon for block-oriented input/output
lockdclnt	Daemon for Network Lock Manager (NLM) clients
lockdsrv	RPC service for Network Lock Manager (NLM)
mountd	Daemon for mounting remote resources
nfsd	NFS server daemon for input/output
pcnfsd	Daemon for supporting DOS PCs
rpcbind	RPC daemon (portmapper)
statd	Status Monitor for status-based RPC services

4.2.1 biod - NFS client daemon for block-oriented input/output

biod is a client daemon for asynchronous block-oriented input/output. This daemon processes read and write jobs (read-ahead, write-behind) on an NFS client.

By default, four *biod* daemons are invoked automatically after starting NFS.

Authorization: The *biod* daemon can only be started with root authorization.

Path: /usr/lib/nfs

Syntax

```
biod[ nservers]
```

`nservers`

Number of daemons to be started. The default for *nservers* is four.

4.2.2 lockdclnt - Daemon for NLM clients

Together with the *lockdsrv* daemon, *lockdclnt* implements the Network Lock Manager (NLM). NLM is a service that provides support for working with file locks via NFS. *lockdclnt* must run on every NLM client computer. It fulfills the following tasks:

- When the client issues its first NLM request, *lockdclnt* determines the address of the NLM server. This address is saved by the client so that it need only be queried again with *lockdclnt* in exceptional cases.
- *lockdclnt* registers the NLM server with the status daemon *statd*.
- *lockdclnt* records which NLM server is accessed by the client. The list of NLM servers is periodically compared with the mount table */etc/mnttab*: If an entry for a mounted NFS resource is deleted from the mount table, *lockdclnt* informs the *statd* status daemon that the associated NLM server no longer needs to be monitored.
- *lockdclnt* establishes an RPC service that is called by *statd* when the client restarts following a crash. This server again sets the client locks on the server.

lockdclnt is started automatically when NFS is started. If the daemon is not running, the system log file should be checked for error messages from *lockdclnt*.

Path: /usr/lib/nfs

Syntax

```
lockdclnt
```

4.2.3 lockdsrv - RPC service for NLM (Network Lock Manager)

Together with the *lockdclnt* daemon, *lockdsrv* implements the Network Lock Manager (NLM). NLM is a service that provides support for working with file locks via NFS. *lockdsrv* must be running on every NLM server and every NLM client. It processes NLM requests from remote clients. In addition, *lockdsrv* registers the clients with the *statd* status daemon.

One of the main tasks of *lockdsrv* is to manage blocked attempts to set a file lock. A process attempts to set a lock, this fails, and the process blocks. If a lock that was previously blocked can be obtained on the server, *lockdsrv* notifies the *lockdsrv* process running on the client. This in turn ensures that the client process which was blocked because it could not set the lock, can now continue.

During initialization, *lockdsrv* uses *statd* to provide information to all NLM clients registered on the server. During a "grace period", which can be defined by the system administrator, the clients can reestablish locks that have been lost on the server as a result of a crash, for example. Any NLM requests other than reestablishing locks will be rejected by *lockdsrv* during this period.

lockdsrv is started automatically when NFS is started. If the daemon is not running, the system log file should be checked for error messages from *lockdsrv*.

Path: /usr/lib/nfs

Syntax

```
lockdsrv[ -g grace_period][ -t retry_timeout][ -f nr_file][ -s]
```

-g grace_period

Grace period in seconds, during which NLM clients can reestablish lost locks.

grace_period has the format: `min:wait:max`

`min`

Minimum total time of grace period.

`wait`

Following each lock reestablishment, *lockdsrv* prolongs the grace period for at least another *wait* seconds, provided the maximum total time of the grace period will not be exceeded.

`max`

Maximum total time of grace period.

The default value for *grace_period* is: `45:10:360`

-t retry_timeout

When locks block, *lockdsrv* attempts to reestablish them all every *retry_timeout* seconds. The default value is 60 seconds.

-f nr_file

nr_file determines the number of files on the server that can be locked by NLM clients. The default value is 1024.

-s

A message is written to the system log file if conflicts arise when establishing file shares (usually by PC clients).

4.2.4 mountd - Daemon for mounting remote resources

mountd is a daemon which responds to requests to mount NFS resources. To do this, it reads the file */etc/dfs/sharetab*. This daemon determines which resources are available for mounting on which computers. It also provides information about which resources are mounted on which clients. This data can be output using the *dfmounts* command.

The *mountd* daemon is invoked automatically after starting NFS.

Authorization: The *mountd* daemon can only be started with root authorization.

Path: /usr/lib/nfs

Syntax

```
mountd
```

Files

/etc/dfs/sharetab

Table of shared resources

4.2.5 nfsd - NFS server daemon for input/output

nfsd is a server daemon. It receives and processes the read and write requests from clients.

By default, four *nfsd* daemons are invoked automatically after starting NFS.

Authorization: The *nfsd* daemon can only be started with root authorization.

Path: /usr/lib/nfs

Syntax

```
nfsd[ -a][ -p protocol][ -t device][nservers]
```

-a

Activates *nfsd* for all available connectionless transport protocols.

-p protocol

Activates *nfsd* for the specified protocol.

-t device

Activates *nfsd* for the transport protocols specified by the given device.

nservers

Number of daemons to be started. *nservers* should be specified according to the load expected on this NFS server. The default for *nservers* is four.

Files

.nfsXXX

Temporary internal file which is created by *nfsd*.

4.2.6 pcnfsd - Daemon for supporting DOS PCs

pcnfsd was a daemon offering support for ONC clients (Open Network Computing) on Windows 95, Windows 98 and Windows NT systems.

It is only delivered for compatibility reasons.

4.2.7 rpcbind - Daemon for RPC

The "portmapper" *rpcbind* is a daemon that converts the RPC program numbers into general addresses (e.g. IP address and port number). It must be running on the server in order to send RPC requests to the server. Because RPCs are used for the NFS-specific network communication, the *rpcbind* daemon is prerequisite for the other daemons.

If a server program is started which communicates via RPC (such as the *mountd* daemon, for example), it informs the *rpcbind daemon* of the address at which it is "listening" and of which RPC program numbers it is capable of processing. If a client wishes to send an RPC request to a specified program number, it first contacts the *rpcbind daemon* on the server computer to determine the address to which it should send the RPC packets.

The *rpcbind* daemon is invoked automatically by an RC script when POSIX is started.

Authorization: The *rpcbind* daemon can only be started with root authorization.

Path: /usr/sbin

Syntax

```
rpcbind
```

4.2.8 statd - Status Monitor for status-based RPC services

The *statd* daemon implements the Network Status Monitor (NSM). It is needed by the Lock Manager to restore lock tables following system crashes, and must run on both the client and the server. *statd* works in conjunction with status-based RPC daemons to provide restart functions following system crashes.

Path: /usr/lib/nfs

Syntax

```
statd
```

Files

/etc/sm

Contains the names of the clients that have locked one or more files.

/etc/sm.bak

Contains the names of the clients that must be notified of a server restart.

4.3 rpcinfo program

The RPC mechanism (remote procedure call) is based on the client-server model. A server offers services and lets the RPC daemon know at which address it is waiting for RPCs from clients and via which protocols communication can be carried out with it. The client sends an RPC to use the server's service over the network. It uses the RPC daemon to contact the server. The following four values are of significance in this situation:

- program number
- version number
- procedure number
- protocol

The present section contains a description of the rpcinfo program.

4.3.1 rpcinfo - Output RPC information

rpcinfo makes an RPC call to an *rpcbind* daemon (portmapper) and reports the result.

If the *rpcinfo* command is specified with the option *-p*, all the RPC services are listed which are registered with the *rpcbind* daemon.

If the *rpcinfo* command is specified with the option *-T*, *rpcinfo* makes an RPC call to the procedure 0 of *program* and *version* on the specified *host* and reports whether a response was received. *transport* is the transport route which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote *rpcbind* daemon.

Path: /usr/bin

Syntax

```
rpcinfo[ host]
```

```
rpcinfo -p[ host]
```

```
rpcinfo -T transport host program version
```

```
rpcinfo[ -n portnum] -u host program version
```

```
rpcinfo[ -n portnum] -t host program version
```

```
rpcinfo -a serv_address -T transport program[ version]
```

```
rpcinfo -d[ -T transport] program version
```

`host`

Name of a remote computer. The default host is the local host. If you specify the *rpcinfo* command with *host*, the RPC services registered with *rpcbind* on *host* are listed.

-T `transport`

Specifies the transport route on which the service is required. If this option is not specified, *rpcinfo* uses the transport route specified in the environment variable *NEPATH* or, if the latter is not set or has a null value, in the network configuration database. This is a generic option, and can be used in conjunction with any other option.

-a serv_address

This uses *serv_address* as the (universal) address for the service on *transport* in order to perform a status check with the *ping* command on procedure 0 of the specified *program* and to report whether a response was received. The *-a* option can only be used with the *-T* option.

If the version number is not specified, *rpcinfo* attempts with the *ping* command to find out all the available version numbers for this program number. This option avoids calls to *rpcbind* on remote computers to locate the address of the service. The *serv_address* has the format of the universal address of the given transport route.

-d

Deletes the registration for the RPC service of the specified *program* and the specified *version*. If *transport* is specified, the service is unregistered only on that transport path. Otherwise, the RPC service is unregistered on all transport routes on which it was registered. This option can only be used by a privileged user.

-n portnum

Uses *portnum* as the port number for the options *-t* and *-u* instead of the port number given by the *rpcbind*. Use of this option avoids a call to the remote *rpcbind* to find out the address of the service.

-p

Probes the *rpcbind* on *host* and outputs a list of all registered RPC programs. If *host* is not specified, the local host is the default host.

-t

Makes an RPC call to procedure 0 of *program* on the specified *host* using TCP and reports whether a response was received.

-u

Makes an RPC call to procedure 0 of *program* on the specified *host* using UDP and reports whether a response was received.

program

Program number, given as a number.

Program name or alias name according to the configuration file */etc/rpc*.

version

If a *version* is specified, *rpcinfo* attempts to call that *version* of the specified *program*. Otherwise, *rpcinfo* attempts to find all registered version numbers for the specified *program* by calling version 0, which is presumed not to exist; if it does exist, *rpcinfo* attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for the option *-d*.

Examples

Example 1:

Show all of the RPC services registered on the local computer:

```
$ rpcinfo
   program version netid address    service owner
    100000    3     udp  0.0.0.0.0.111  portmapper  superuser
    100000    2     udp  0.0.0.0.0.111  portmapper  superuser
    100000    3     tcp  0.0.0.0.0.111  portmapper  superuser
    100000    2     tcp  0.0.0.0.0.111  portmapper  superuser
    ...
    100003    2     udp  0.0.0.0.8.1  nfs  superuser
    100003    3     udp  0.0.0.0.8.1  nfs  superuser
    ...
    100005    1     udp  0.0.0.0.20.177  mountd  superuser
    100005    1     tcp  0.0.0.0.20.178  mountd  superuser
    100005    3     udp  0.0.0.0.20.177  mountd  superuser
    100005    3     tcp  0.0.0.0.20.178  mountd  superuser
    ...
```

Example 2:

Show all of the RPC services registered with *rpcbind* on the computer *klaxon*:

```
$ rpcinfo klaxon
...
```

Example 3:

Show whether the RPC service with program number 200007 and version 3 is registered on the computer *klaxon* for the transport route *tcp*:

```
$ rpcinfo -T tcp klaxon 200007 3
program 200007 version 3 ready and waiting
```

Example 4:

Show all of the RPC services registered with the rpcbind on the local computer:

```
$ rpcinfo -p
   program vers proto   port
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100005    1   udp   5297  mountd
    100005    1   tcp   5298  mountd
    100005    3   udp   5297  mountd
    100005    3   tcp   5298  mountd
    150001    1   udp    603  pcnfsd
    100024    1   tcp   5301  status
    100024    1   udp   5302  status
    100021    1   tcp   5304  async-nlockmgr
    100021    1   udp   5305  async-nlockmgr
    100021    2   tcp   5304  async-nlockmgr
    100021    2   udp   5305  async-nlockmgr
    100021    3   tcp   5304  async-nlockmgr
    100021    3   udp   5305  async-nlockmgr
    100021    4   tcp   5304  async-nlockmgr
    100021    4   udp   5305  async-nlockmgr
```

Example 5:

Check the status of nfs (program number 100003) on the host bs2host1:

```
$ egrep '^nfs' /etc/rpc
nfs             100003  nfsprog
$ rpcinfo -T udp bs2host1 nfs
program 100003 version 2 ready and waiting
program 100003 version 3 ready and waiting
```

Example 6:

Delete the registration for version 1 of the walld service (program number 100008) for all transport routes:

```
$ rpcinfo -d 100008 1
```

4.4 Administration files

This section describes the administration files.

```
/
`-- etc
    |-- dfs
    |   |-- dfstab
    |   |-- fstypes
    |   `-- sharetab
    |-- mnttab
    |-- print
    |-- rmtab
    |-- rpc
    `-- vfstab
```

Figure 7: Administration files

Certain administration files are used for the operation of NFS. They are created automatically either by NFS or by POSIX at startup time. The files are used for the following administration functions:

- The files */etc/print*, */etc/rpc*, */etc/vfstab*, */etc/dfs/dfstab* and */etc/dfs/fstypes* are used for the automatic management of resources. Default values are entered in these files, values which are interpreted by the commands for the purpose of NFS resource management.
- The files */etc/dfs/sharetab*, */etc/mnttab* and */etc/rmtab* are used for information about resources. The commands entered for managing NFS resources log their actions in these files.

The files can only be modified with root authorization. To do so, call the editor *edtu* or *vi* in the POSIX shell. See the "POSIX Commands" manual.

The following overview lists the administration files:

File	Function
/etc/mnttab	Table of mounted file systems
/etc/print	Templates for BS2000 print commands
/etc/rmtab	Table of mounted remote resources
/etc/rpc	RPC program number file
/etc/vfstab	Table of defined file systems
/etc/dfs/dfstab	Table of resources to be shared
/etc/dfs/fstypes	Table of installed utilities for distributed file systems
/etc/dfs/sharetab	Table of shared resources

Table 5: Administration files

4.4.1 /etc/mnttab - Table of mounted file systems

The file */etc/mnttab* contains information about all the file systems mounted on the local computer. This file contains information which is generated by the *mount* command.

Each line contains the following information; items are separated by any number of blanks and/or tabs:

Format

```
resource     mountp    fstyp    spec_options    time
```

`resource`

Absolute path name of the mounted file system. Remote resources have the form: *server:pathname* where *server* is the computer name of the NFS server making the resource available and *pathname* is the absolute path name of the resource.

`mountp`

Absolute path name of the mount point.

`fstype`

File system type.

`spec_options`

Options as specified for the *mount* command.

`time`

Mount time, given in seconds since 1.1.1970

Example

```
$ cat /etc/mnttab
/dev/root      /            ufs    rw,suid          1686653910
/proc          /proc        proc   rw,              1686653911
/dev/fd        /dev/fd      fdfs   rw               1686653911
/dev/dsk/3     /var         ufs    suid,rw,noquota  1686653911
/dev/dsk/2     /home1       ufs    suid,rw,noquota  1686653911
SINTEST1:/nfs  /nfsclient   ufs    rw               1687246196
```

4.4.2 /etc/rmtab - Table of mounted remote resources

The file */etc/rmtab* contains details about all the remote resources mounted on the local computer. This file contains information which is generated by the command *mount -F nfs*.

Each line contains the following information:

Format

```
resource
```

`resource`

Absolute path name of the mounted file system of type *nfs*. The specification has the form: *server: pathname* where *server* is the computer name of the NFS server making the resource available and *pathname* is the absolute path name of the resource.

Example

```
$ cat /etc/rmtab
SINTEST1:/nfs1/nfsserver
```

4.4.3 /etc/rpc - RPC program number file

The RPC program number file contains user-readable names which can be used internally instead of RPC program numbers.

Each line contains the following information; items are separated by any number of blanks and/or tabs:

Format

```
programname     programnumber    aliases    # text
```

`programname`

Name of the RPC server program

`programnumber`

RPC program number

`aliases`

Alias names which can be used instead of the program number

`# text`

Comment. The special character # indicates the start of a comment.The comment ends at the end of the line.

Example

```
$ egrep '100000|100003|100005|150001|100024|100021' /etc/rpc
async-nlockmgr  100021
status          100024
portmapper      100000  portmap sunrpc
nfs             100003  nfsprog
mountd          100005  mount showmount
pcnfsd          150001  pcnfs
```

4.4.4 /etc/vfstab - Table of defined file systems

The file */etc/vfstab* describes default values for each file system defined on the local computer. The file can be edited using *edt* or *vi*.

The file systems which are listed in the file */etc/vfstab* with *automnt=yes*, are mounted on starting POSIX or if the *mountall* command is entered without specifying a file name (or "-"). File systems of type *nfs* are mounted by means of *mountall -F nfs* when starting NFS.

The fields in the table are separated by blanks. A hyphen (-) indicates a blank entry in the respective field. The table contains the following fields:

Format

```
special  fsckdev  mountp  fstype  fsckpass  automnt  mntopts
```

`special`

Describes the resource to be mounted. For file systems of type *nfs*, *special* has the following form: *server: pathname* where *server* is the computer name of the NFS server making the resource available and *pathname* is the absolute path name of the resource.

`fsckdev`

Name of the character-oriented device. When mounting remote resources, this parameter is not supported and must be a hyphen (-).

`mountp`

Mount point. Absolute path name of the directory in which the resource is to be mounted.

`fstype`

File system type. For remote resources *nfs* must be entered.

`fsckpass`

The pass number to be used for multiple *fsck* commands. When mounting remote resources, this parameter is not supported and must be a hyphen (-).

`automnt`

Specifies whether (*yes*) or not (*no*) the resource is to be mounted automatically by *mountall* at POSIX startup time. File systems of type *nfs* are not mounted at POSIX startup time but by the command *mountall -F nfs* at NFS startup.

`mntopts`

List of options separated by commas for mounting the file system. The options are the same as the *specific_options* of the *mount* command.

Example

```
$ cat /etc/vfstab
/dev/root       /dev/rroot      /               ufs     1       yes     -
/proc   -       /proc   proc    -               no      -
/dev/fd -       /dev/fd fdfs    -               no      -
/dev/dsk/3      /dev/rdsk/3     /var            ufs     1       yes     -
/dev/dsk/2      /dev/rdsk/2     /km1            ufs     1       yes     -
/dev/dsk/4      /dev/rdsk/4     /nfs1           ufs     1       yes     -
/dev/dsk/5      /dev/rdsk/5     /tmp1           ufs     1       yes     -
/dev/dsk/6      /dev/rdsk/6     /vsx            ufs     1       no      -
/dev/dsk/7      /dev/rdsk/7     /vsx-ro         ufs     1       no      -
/dev/dsk/8      /dev/rdsk/8     /usr1           ufs     1       yes     -
/dev/dsk/9      /dev/rdsk/9     /usr2           ufs     1       yes     -
/dev/dsk/10     /dev/rdsk/10    /home1          ufs     1       no      -
/dev/dsk/11     /dev/rdsk/11    /home2          ufs     1       no      -
/dev/dsk/12     /dev/rdsk/12    /home3          ufs     1       no      -
/dev/dsk/13     /dev/rdsk/13    /home4          ufs     1       no      -
/dev/dsk/14     /dev/rdsk/14    /home5          ufs     1       no      -
/dev/dsk/15     /dev/rdsk/15    /home6          ufs     1       no      -
/dev/dsk/16     /dev/rdsk/16    /oldroot        ufs     1       no      -
/dev/dsk/17     /dev/rdsk/17    /ascii          ufs     1       yes     -
tanz:/usr/local -               /usr/local/tmp  nfs     -       yes     ro
```

4.4.5 /etc/dfs/dfstab - Table of resources to be shared

The file */etc/dfs/dfstab* contains *share* commands for making resources available. The *share* commands which are listed in the file */etc/dfs/dfstab* are executed when the *shareall* command is entered with no input file specification.

Format

```
share[ -F nfs][ -o specific_options][ -d description][ pathname]
```

The format of the lines in this file corresponds to the syntax of the *share* command.

4.4.6 /etc/dfs/fstypes - Table of installed utilities for distributed file systems

The file */etc/dfs/fstypes* contains a list of the utility packages, installed on the system, for distributed file systems.

When NFS commands are started without the option *-F nfs*, the system uses the file systemtype listed in the first line of this file as the default.

Format

```
fstype designation: version
```

`fstype`

File system type

`designation`

Name of the utility package

`version`

Version of the utility package

Example

If NFS V3.0 is the only utility package for distributed file systems installed on the system, this file contains the following as its first and only line:

```
nfs   Network File System Utilities: 3.0
```

4.4.7 /etc/dfs/sharetab - Table of shared resources

The file */etc/dfs/sharetab* contains a table of the local resources which have been made available for client access by means of the *share* command.

Each line in the table contains the following fields. The fields in the table are separated by blanks. A hyphen (-) indicates a blank entry in the respective field:

Format

```
pathname  resource  fstype  specific_options  description
```

`pathname`

Path name of the resource made available.

`resource`

Symbolic name via which remote computers can access the resource.

`fstype`

File system type of the resource made available.

`specific_options`

The file system specific option specified when the resource was shared.

`description`

Comment describing the resource made available.

Examples

```
$ cat /etc/sharetab
/export/graphics        -       nfs     ro,rw=art.dept
```

5 Troubleshooting, performance enhancement

This chapter gives an insight into internal operations, describes problems which may occur under certain circumstances while you are using NFS, and explains performance enhancement measures.

The technical details contained in this chapter are intended to facilitate troubleshooting for an experienced system administrator. To learn about NFS in more detail you should refer to the publication "Managing NFS and NIS".

5.1 The mount operation summarized

This section describes the internal operations when NFS is started and when a remote resource is mounted.

Starting NFS

The NFS daemons are started resp. restarted by running the script *"/etc/rc2.d/S20nfs start"* or by sending the signal SIGTRAP to the init process (PID 1). This is done automatically at POSIX subsystem start or at NFS (update) installation.

By default following daemons are started:

```
1 mountd
4 nfsd
4 biod
1 pcnfsd
1 rpcbind
```

Mounting a remote resource

As an example for all mount operations involving remote resources, a description follows of what happens when the command *mount -F nfs ...* is entered.

1. You enter the following command:
   `mount -F nfs -o intr,rw,soft tanz:/usr/src /usr/src/tanz.src`

2. A check is made whether in the directory */etc/fs/nfs* the *mount* command for mounting remote resources is present and whether the entered command is syntactically correct.

3. The *mount* command checks in the file */etc/mnttab* whether the resource is already mounted.

4. Via the *rpcbind* daemon, the *mount* command requests the port number which the *mountd* daemon has on the computer *tanz*.

5. The *mount* command passes the path name of the resource (*/usr/src*) to the *mountd* daemon on the computer *tanz*.

6. The *mountd* daemon on the computer *tanz* processes the mount request and checks whether the resource has been shared and with what permissions.

7. The *mount* command receives a positive response from the *mountd* daemon on the computer *tanz* and adds an entry for the mounted resource to the file */etc/mnttab*.

8. When a file from the mounted resource is to be processed, the request is processed by the *nfsd* daemon on the computer *tanz*. Using the options specified when sharing the resource, this daemon checks whether the request can be satisfied or must be rejected.

5.2 Troubleshooting

The ability to eliminate NFS problems presupposes a thorough understanding of the circumstances under which they can occur. When locating an NFS problem, it is always important to be aware that one of three main components can cause it:

- server
- client
- network

It is therefore recommended to look at these three components separately when troubleshooting, so that the cause of the problem can be localized more easily.

5.2.1 Server problems

The messages concerning NFS server problems are output on the BS2000 console. They consist of a POSIX message which contains the NFS problem description as insert *&00*:

```
POS1020 Message of the POSIX kernel: &00
```

The problems described in the following occur when accessing remote resources. The access can be effected by commands or from programs.

When accessing server resources, a distinction is made between "hard" and "soft" mounted directories (see the *hard* and *soft* options of the *mount* command).

If a resource has been "hard" mounted and the server is unavailable for some reason, all programs accessing this resource are blocked. The NFS client issues the following console message:

```
POS1020 ...: NFS server hostname not responding, still trying
```

As soon as the server is accessible again, the following console message appears:

```
POS1020 ...: NFS server hostname ok
```

It is advisable in this case to mount resources with additional use of the option *-o intr* in order to allow blocked programs to be canceled.

If a resource has been "soft" mounted and the server is unavailable, the following console message is issued (possibly multiple times):

```
POS9999 rfscall: function: RPC_TIMEDOUT retry #n: timeo=...
```

If the retries should not succeed, this is reported by the console message:

```
POS1020 ...: NFS function failed for server hostname: RPC: Timed out
```

In this case you should first check whether the server is active and accessible. Enter the following command, where *hostname* is the name of the NFS server:

```
$ /usr/bin/rpcinfo -p hostname
```

If the server is active, a list of the programs running on the server is displayed, with version numbers, protocols and port numbers. If no list is displayed, you should check whether the *rpcbind* daemon is running on the server.

If the *rpcbind* daemon is active on the server but not accessible for the client, you should check the network connection between server and client. On a BS2000 computer with POSIX this can be done in two ways:

- in the POSIX shell with: `ping hostname`
- in BS2000 command mode with: `PING4 hostname` or `PING6 hostname`

When the server is available again, the resource must be remounted.

5.2.2 Client problems

This section deals with problems which can occur during the mounting of NFS resources. Each individual step of the mounting process can lead to an error - some can even cause several errors. The possible causes are given for each of the error messages listed in the following. The messages which NFS issues relating to client problems are output to *stdout*.

In the following examples it is assumed that the resource is mounted by way of the command line, but the troubleshooting methods described also apply to automatic mounting via the file */etc/vfstab*.

`nfs mount: server:pathname: server not responding: RPC: program not registered`

The server which is providing the resource to be mounted is not active.
The *rpcbind* daemon is not running at the server.
The network connection to the server is faulty.

`nfs mount: server:pathname: remote mount server not responding: RPC: Timed out`

Although the *mount* command was able to access the *rpcbind* daemon on the server, the NFS daemon *mountd* is not reachable there.
Check whether the NFS daemons are active on the server.

`nfs_mount: server:pathname: server not responding: RPC: program unavailable`

Check whether NFS is installed on the server.

`mount: mount-point does not exist`

The local mount point does not exist.

`mount: ...: Not a directory`

The local mount point is not a directory.

`nfs_mount: access denied for server:pathname`

The resource has not been shared or the name of your system is not contained in the list of clients authorized for access. Check whether and how the resource to be mounted is entered in the table of shared resources on the server. Do this by entering the following command on the client, where *server* is the name of the NFS server:

```
$ showmount -e server
```

If the resource is not displayed, then it must be made available on the server. Do this by entering the following command on the server, where *pathname* is the name of the NFS resource:

```
$ share -F nfs [-o ...] pathname
```

`sh: file: cannot create`

The user does not have the required access rights or user number.
The resource is shared read-only or mounted read-only.

5.3 Performance enhancement measures

Performance problems with NFS may be attributable to various causes:

- Insufficient disk performance on the server can limit the throughput rate at which reads and writes can be performed.
- Excessive CPU load on client or server limits their capability to service network requests.
- A congested network can limit the transfer rate or increase the retry rate for requests or data transfers.

Improving read and write access

If the data transfer for an NFS read or write access was not successfully completed, then the entire data block is transferred once again. If the retry rate is too high, the size of the NFS read or write blocks must be reduced. This can be done by setting the option *rsize* and *wsize* in the file */etc/vfstab* or, during mounting, by using the *mount* command.

Try choosing the value 2048 bytes or 1024 bytes for the entry in the file */etc/vfstab*:

```
bobserver:/home/bob - /home/bob nfs - - rw,rsize=2048,wsize=2048
```

Since a reduction in size of the read and write blocks restricts the maximum possible performance, it may also be advantageous to reduce the time limit for the request retries. Set the option *timeo* for the *mount* command or in the file */etc/vfstab* to 8 or 6 tenths of a second (the default is 11 tenths of a second).

Improving network performance

If a mounted NFS file system extends over a wide network encompassing many gateways or great distances between clients and server, it may be necessary to increase the option *timeo* for the *mount* command. Slow network hardware can cause long delays in the transmission path. The time limit may need to be increased in this case. For example, set *timeo* to 50 or 100 tenths of a second.

Use *nfsstat -rc* to check the number of NFS/RPC transfer retries and errors affecting the client.

6 Glossary

client

In conjunction with NFS, a client is the computer which accesses resources that have been made available, or “shared”, by another computer (server).

container file

Physical storage medium for a file system in the POSIX file tree. A container file is a PAM file which is located on a PVS (public volume set).

daemon

Daemons are system processes which run permanently and normally in background mode, and which perform general tasks.

directory

A directory is used to group and organize files and subordinate directories of a hierarchical file systems.

file lock

Processes that work on shared files use locks to synchronize accesses to these files. A separate protocol known as "NLM" (Network Lock Manager) is implemented for working with locks in NFS.

file system

A file system is a hierarchical group of directories and files which are located physically on the same storage medium, e.g. in a partition or in a container file.The term is used for organizational structures of files, such as UNIX file system, POSIX file system, hierarchical file system, other BS2000 file systems (DMS and LMS), for example.

file system type

Type of a file system in the file tree of POSIX or of a UNIX computer. The most familiar types are as follows:
Type *ufs*: local file systems containing user data.
Type *nfs*: file systems which are located physically on remote computers.
Examples of further types: *fdfs*, *proc*, *ext3*, *bs2fs*

file tree

Overall hierarchy of the files on a UNIX computer or in POSIX. The UNIX or POSIX file hierarchy is organized on the basis of a tree structure. The root of the file tree is the root directory (/). All other directories are branches which emanate from the root. The files are the leaves of the tree.

heterogeneous system environment

A computer network in which computers from different manufacturers and with different operating systems communicate with each other. Synonym: open computer network.

IP (Internet Protocol)

The Internet Protocol is a protocol which selects the route in a computer network. It performs some of the functions required by Layer 3 of the OSI Reference Model. The TCP and UDP protocols are based on IP.

LAN (local area network)

A LAN is a computer network which is limited to a certain physical area. A LAN can be linked with other computer networks as a private sub network, thus forming a part of a larger network, for example a WAN. Synonyms: local computer network, local network.

local computer

The computer on which the user works directly is referred to as the local computer.

mounting

Logical, non-physical adoption of a remote resource into the local file tree.

OSI Reference Model

The model for communication between open systems, the OSI Reference Model (Open System Interconnection), provides the basis for ISO standardization of data communications. The OSI model structures the organization of communications systems and provides the basis for standardization of the protocols and services. It defines which functions must be implemented by the components involved in the communications.

The OSI Reference Model consists of seven hierarchically organized layers. Specific functions within the framework of the overall communications task are assigned to each layer.

port number

A port number allows to address a particular application within a computer. It corresponds to the address of an application in a computer. The combination of Internet address and port number uniquely identifies the receiver or sender of a data packet within the network.

POSIX file system

File system on a BS2000 computer running POSIX. The POSIX file system corresponds to a UNIX file system.

PVS (public volume set)

A logical group of public (not private) BS2000 volumes. Also referred to as a pubset.

remote computer

A computer in a computer network which a user does not work with directly. The computer on which the user works directly is known as the local computer. Users can communicate with remote computers in the network.

resource

Files or directories which are used with NFS.

root directory

The directory at which the file tree begins. The root directory is represented by the slash (/).

RPC (remote procedure call)

The remote procedure call is a method which is used for communication between client and server applications in distributed processing. This method is used by NFS. A server program is identified by means of a program number, a procedure number and a version number.

RPC is based on the TCP and UDP protocols. Only UDP is used by NFS in BS2000.

server

In conjunction with NFS, a server is the computer which makes resources available that can be mounted and processed by other computers (clients).

sharing

Making local resources available for mounting on remote computers; also referred to as exporting.

TCP (Transmission Control Protocol)

TCP is a connection-oriented protocol which handles data transport between two computers. Unlike UDP, TCP provides secure data transfer (end-to-end connection) and belongs to Layer 4 of the OSI Reference Model.

transparent file access

The user can access files on a remote machine like files on the local machine.

UDP (User Datagram Protocol)

UDP is a connectionless protocol which handles data transport between two computers. UDP may be placed approximately on Layer 4 of the OSI Reference Model. UDP is a datagram protocol which supports broadcasting. In contrast to TCP (secured end-to-end protocol), UDP only guarantees that the message was sent successfully.

UFS (UNIX file system)

Local file system for UNIX.

WAN (wide area network)

A WAN is a computer network which is not restricted to a spatially limited area.

7 Related publications

openNet Server BCAM
User Guide

BS2000 System Administration
User Guide

BS2000 Commands
User Guide

EDT Statements
User Guide

POSIX Basics
User Guide

POSIX Commands
User Guide

POSIX BS2000 file system bs2fs
User Guide

Managing NFS and NIS
Hal Stern, Mike Eisler and Ricardo Labiaga
Second edition, published June 2001
O‘Reilly & Associates, Inc
ISBN 1-56592-510-6