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/OSD 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 (V15.0 or later) must be installed.

NFS V1.2C runs on BS2000/OSD-BC V2.0 or later and requires POSIX-BC, which is a component of BS2000/OSD-BC (V2.0 or later).

NFS V3.0 runs on BS2000/OSD-BC V3.0 or later and 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 the manual “POSIX Basics” in your possession.

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 describes how to connect up a PC and make use of the print services of a BS2000 from the PC.

Chapter 6 gives advice on troubleshooting and error recovery, along with measures you can undertake to enhance the performance of NFS.

The manual also contains a glossary, a list of related publications and an index.

1.4 README file

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README file. You will find the README file on your BS2000 computer under the file name SYSRME.*product.version.language*. The user ID under which the README file is cataloged can be obtained from your system administrator. You can view the README file using the /SHOW-FILE command or an editor, or print it out on a standard printer using the following command:

```
/PRINT-DOCUMENT filename, LINE-SPACING=*BY-EBCDIC-CONTROL
```

with SPOOL versions lower than 3.0A:

```
/PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=
    PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```

1.5 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

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

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

[ ] 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

`bold typewriter face`

: Inputs: 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.

`normal typewriter face`

: Outputs

1.6 Changes compared to the previous manual

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

– In addition to NFS protocol version 2, NFS V3.0 also supports NFS protocol version 3.

– NFS V3.0 provides full functionality for locking files. The daemons *lockd-srv*, *lockd-clnt* and the Status Monitor *statd* are available for this purpose.

– NFS V3.0 supports 64-bit systems. This makes it possible to mount file systems containing files larger than 2 GB.

– NFS V3.0 supports asynchronous write mode (`commit`).

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.

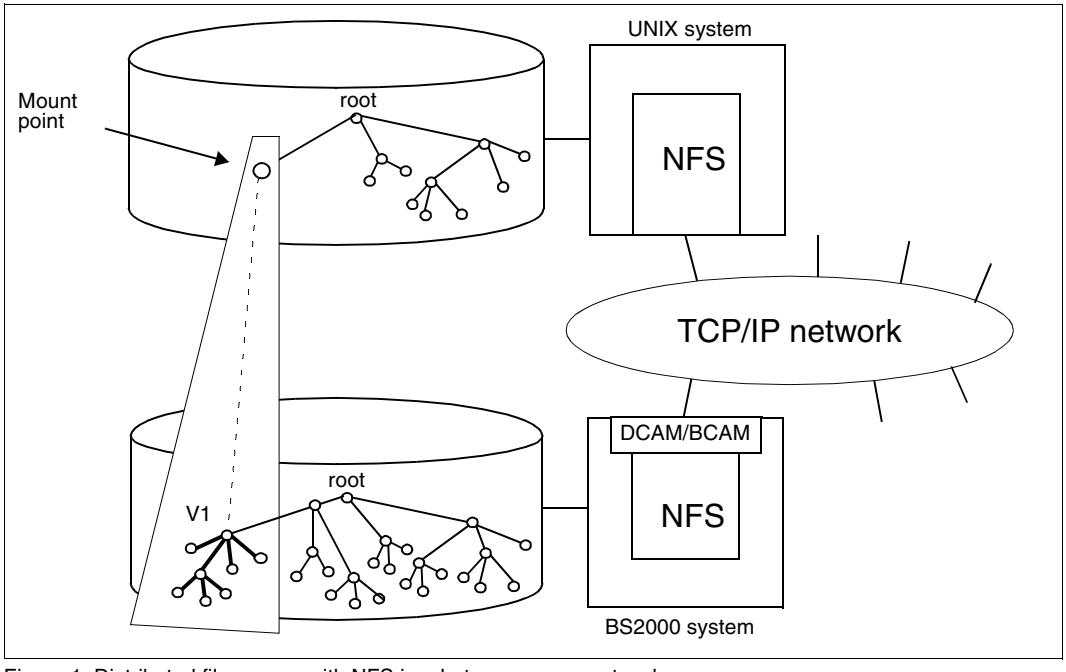


Figure 1: Distributed file access with NFS in a heterogeneous network

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 serves 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 designated as an NFS server if it makes available local resources for remote systems.
– a system is designated as 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 it 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 hirarchy.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 relevant NFS server.

2.1.2 NFS versions and NFS protocol versions

The product version NFS V1.2C, supports the NFS protocol version 2. With the product version 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).

All NFS versions are based on the UDP protocol (User Datagram Protocol). This is a connectionless (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.

Moreover, NFS V3.0 also allows you to mount file systems containing files larger than 2 GB. In order to work with such files, X/Open has written an application interface that facilitates the transition to large files and is supported with CRTE V2.2B (see also the section “File systems larger than 2 Gbytes” on page 11).

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 resends 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 *lockd-srv* 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 *lockd-clnt* 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, clear-up 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 reset 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 (*lockd-srv / lockd clnt*) is the only service that uses the NSM (*statd*) (see also the description of *lockd-srv* and *lockd-clnt* in the section “Daemons” on page 64).

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 by the Monitor of the restart. 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” on page 64.

2.1.6 File systems 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). With the next POSIX version, it will be possible to set up large file systems locally and make them available to NFS clients.

Support for large files stretches to all accesses where file areas are addressed, including 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 have 64-bit capability, large files can be processed without restriction. However, problems arise if the client and server have different word lengths.

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. As with the behavior of 32-bit applications on local 64-bit file systems, system calls from the client may fail in such cases.

Exception:

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

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 differentiate between a 32-bit and a 64-bit client. 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/OSD

NFS in BS2000/OSD belongs to the POSIX program packages. The prerequisites for the use of NFS in BS2000/OSD and for the interoperation of NFS and POSIX are described in the following.

2.2.1 Network connection

Refer here also to the manuals "BCAM, Volume 1 and Volume 2".

A network with a TCP/IP communications capability is prerequisite for working with NFS. For this you will need the *open*Net Server product. BCAM (V12 or later) is required to connect a BS2000 computer to a TCP/IP network. BCAM is a component of *open*Net 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/OSD.

2.2.2 POSIX

Refer here also to the manual "POSIX Basics".

In BS2000/OSD V2.0 and higher, 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, Volume 1 and Volume 2. POSIX in BS2000 is supplied in two products:

– POSIX-BC is a component of the basic configuration of BS2000/OSD V2.0 and contains the POSIX shell with a basic complement of POSIX commands (also called the basic shell), the C library functions 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 jobs of privileged and non-privileged users. It performs the tasks of the UNIX system kernel in BS2000. It comprises:
– a UNIX system kernel which has been ported into BS2000
– BS2000 interfaces and services which establish a link between the ported UNIX system kernel and BS2000
– routines for initializing and terminating the POSIX subsystem

C library functions

CRTE V2.0 and higher 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

POSIX means that a further file system is available in BS2000: the POSIX file system. This corresponds to a UNIX file system. NFS works with this file system.

The POSIX installation program creates a POSIX file system which 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.

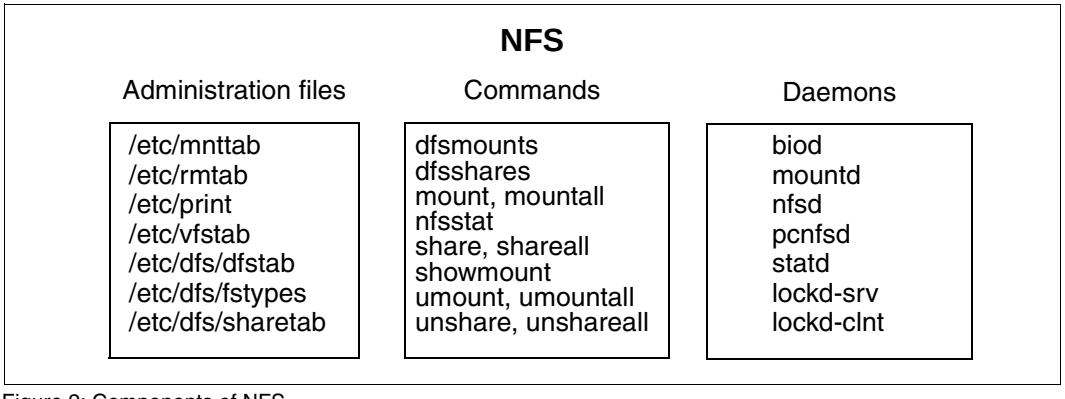


Figure 2: Components of NFS

Commands

NFS offers commands which serve to make available (share) local resources and to mount remote resources, and also to 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. They also support the PC connection capability. 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

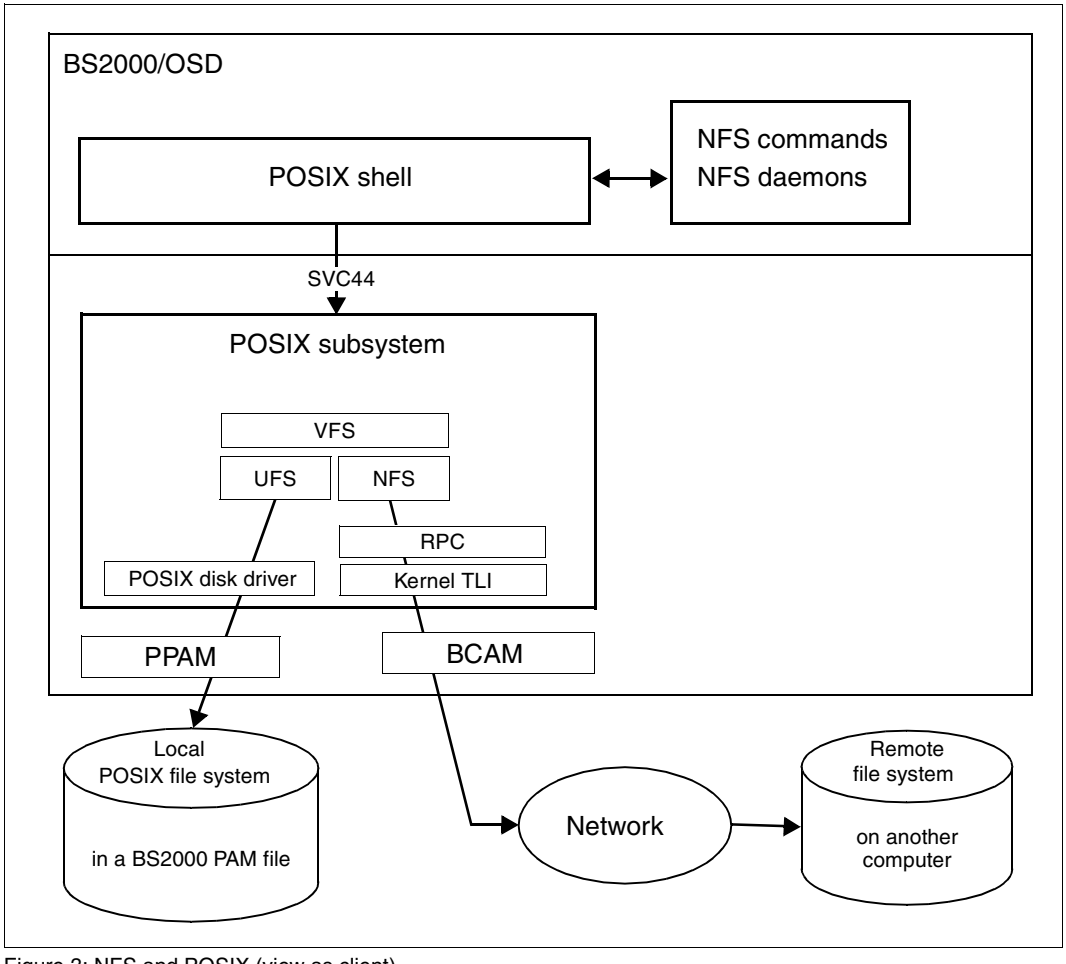


Figure 3: NFS and POSIX (view as client)

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

- The NFS commands are entered in the POSIX shell.
- Remote file system requests, amongst other things, 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 product PPAM (Privileged Primary Access Method).

– If a remote resource is to be processed, the request is handled via NFS. Communication takes place with the remote computer via RPCs (remote procedure calls) along with Kernel TLI (Transport Layer Interface) and the BS2000 product 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 here 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 allocate the POSIX user number (uid):

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

For POSIX group administration and the administration of accesses via remote computers with *rlogin* there exist further 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 been allocated the POSIX user number (uid) 0 and the POSIX group number (gid) 0. Root authorization is required for starting NFS and for the internal POSIX administration of file systems. Root authorization is allocated to the SYSROOT user ID by default. Root authorization is allocated 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 here 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
- File	r read	r read	r read
b Block-oriented device	w write	w write	w write
c Character-oriented device	x execute	x execute	x execute
d Directory			
l Symbolic reference			
s Execution bit			

Table 1: Protection bits and identification characters

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
– permission to create device files and/or access them

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. Clients wishing to have write access to the resource must be logged on under the same *uid* on their computer as the owner of the resource (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 job. For each client access it checks whether the port number via which the client job arrives is privileged, i.e. is less than 1024. If it is not privileged, the client job 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.040*. Refer here also to “Installing and starting POSIX” on page 26:

`PORTMON=1`	Port monitoring is activated; client accesses via 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 file systems, plus special considerations to be observed 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

Incorporation of the BS2000 computer into a TCP/IP network is prerequisite for the use of NFS. You will need the openNet Server product for this purpose.

Loading the delivery unit files

The files of the NFS delivery unit must be loaded under the standard user ID ($.). This is normally done using the SOLIS procedure. If the IMON (Installations MONitor) product is started on the system, installation can be performed using IMON. As of BS2000/OSD-BC V3.0 and OSD-SVP V2.0, the installation must be carried out with the IMON installation monitor.

The NFS delivery unit comprises the following files.

SINLIB.NFS.030 or SINLIB.NFS.012	Library containing commands, daemons etc.
SYSSII.NFS.030 or SYSSII.NFS.012	IMON information file
SYSFGM.NFS.030.D or SYSFGM.NFS.012.D	Release Notice in German
SYSFGM.NFS.030.E or SYSFGM.NFS.012.E	Release Notice in English

Table 2: The files of the NFS delivery unit

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 (Administrate POSIX filesystems)
– installation of the POSIX program packages (Install packages on POSIX)
– uninstallation 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 system has been set up. Once POSIX has been initially installed, the minimum configuration of a POSIX file system contains the root directory / and the directory /var, each in their own separate container file.

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 file SYSSSI.POSIX-BC.040. The modified parameters take effect the next time the POSIX subsystem is started.

Installing NFS

Authorization: Installing of NFS requires the BS2000 privilege TSOS and Root authorization.

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

                    Install POSIX subsystem

                    Administrate POSIX filesystems

                    Install packages on POSIX

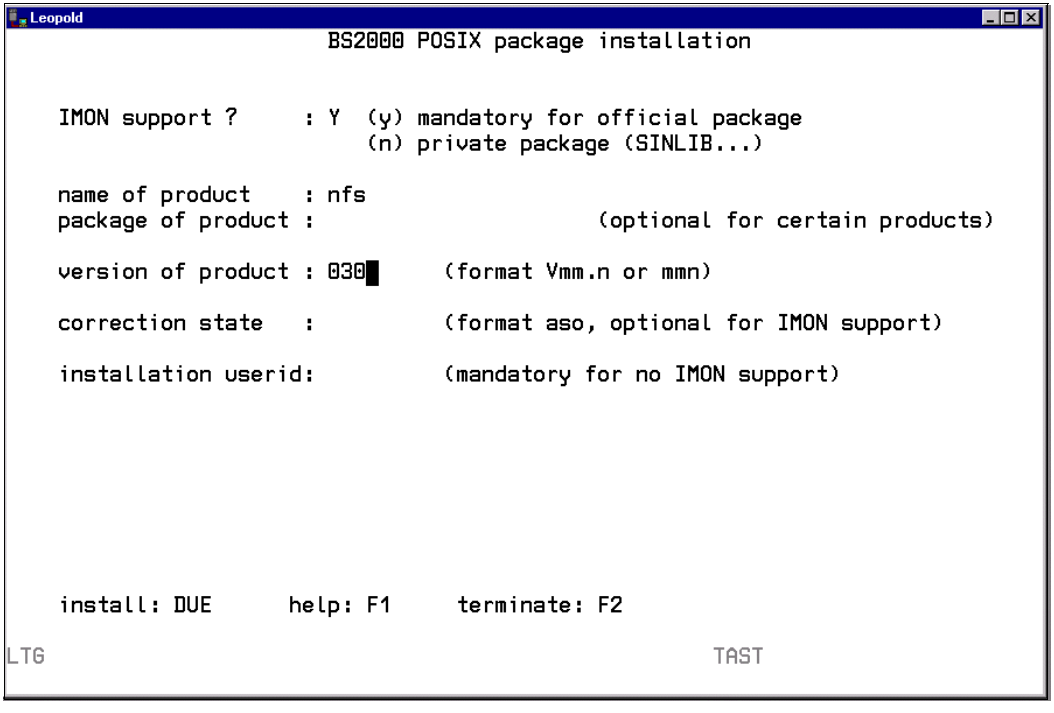
                    Delete packages from POSIX




Select:  MAR + DUE                                  Finish installation: F2
Help  :  F1
```

► Select the menu *Install packages on POSIX*

The following submenu then appears:



- Enter *nfs* for the product and *030* for the version. NFS is installed.

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

Upgrading from previous NFS versions (V1.2B)

- Deinstall the predecessor product using *Delete packages from POSIX.*
- Then reinstall NFS as described above.

NFS V3.0 (or NFS V1.2C) requires correction level A31 of POSIX. If necessary, perform an update installation of POSIX-BC.

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 be called 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 Implementing 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 as required during a server session
- mounting and unmounting resources as required during a working session
- modifying 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 the chapter "Troubleshooting, performance enhancement" on page 99)

3.3.1 Starting and stopping NFS

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

At startup time, all resources entered in the */etc/dfs/dfstab* file with the *-F nfs* option are made available (see the section “Administration files” on page 80).

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 with the SDF command:

  ```
  /START-POSIX-SHELL
  ```

- In the POSIX shell, start NFS with the *S20nfs* script. The *S20nfs* script is located in the */etc/rc2.d* directory. Enter:

  ```
  /etc/rc2.d/S20nfs start
  ```

All daemons are started automatically when NFS is started. The following message is displayed on the screen:

```
*** starting NFS V3.0 ***
```

At startup time, all resources entered in the */etc/dfs/dfstab* file with the *-F nfs* option are made available. After NFS has started, the following command can be entered in order to mount predefined resources:

- `mountall -F nfs`

The *mountall* command serves to mount all resources of type *nfs* which are entered in the file */etc/vfstab*.

Terminating NFS manually

Authorization: Root authorization is required for terminating NFS.

- NFS is terminated with the *K20nfs* script. The *K20nfs* script is located in the directory */etc/rc0.d*. In the POSIX shell, enter:
- `/etc/rc0.d/K20nfs stop`

When NFS is terminated, the availability of resources is revoked, all file systems of type *nfs* are unmounted and all daemons are terminated. The following message is displayed on the screen:

```
*** NFS V3.0 going down ***
```

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: *Administrate POSIX filesystems*. After initial installation of POSIX has been completed, the POSIX file tree in its minimum configuration contains the root directory */* and the directory */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. It cannot subsequently be modified again.

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). In order to achieve enhanced performance, not all container files should be placed on the same disk.

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.

Incorporation of file systems created with NFS V1.0

File systems which were created using NFS V1.0 can be mounted in the POSIX file tree. The mount point must not be either */* (root) or */var*, in other words the file systems */* and */var* created during the initial installation of POSIX must not be hidden by the previous version.

The container files containing the file systems from the previous version which are to be mounted must exist in non-key format. If a container file exists in key format, it must be converted into non-key format using the PAMCONV (BS2000 utility). Once this has been done, the file system it contains can be mounted.

3.3.2.1 Code conversion

Refer here also to the manual “POSIX Commands”, command: iconv

The files of the POSIX file system are handled as standard in EBCDIC format as per EDF03 or EDF04 by POSIX. The POSIX command *iconv* is provided for conversion to ASCII format as per 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 must be set to YES. Use the following command for this purpose:

```
IO_CONVERSION=YES
export IO_CONVERSION
```

Example

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

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

3.3.2.2 BS2000 files

Refer here also to the manuals
“POSIX Basics” and
“POSIX Commands”, command: bs2cp

BS2000 files can be copied into the POSIX file system using the POSIX command *bs2cp* and thus be made available for other systems in the network with 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). Local resources are released 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 *dfstab* file in the */etc/dfs* directory. The *dfstab* 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 released 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.

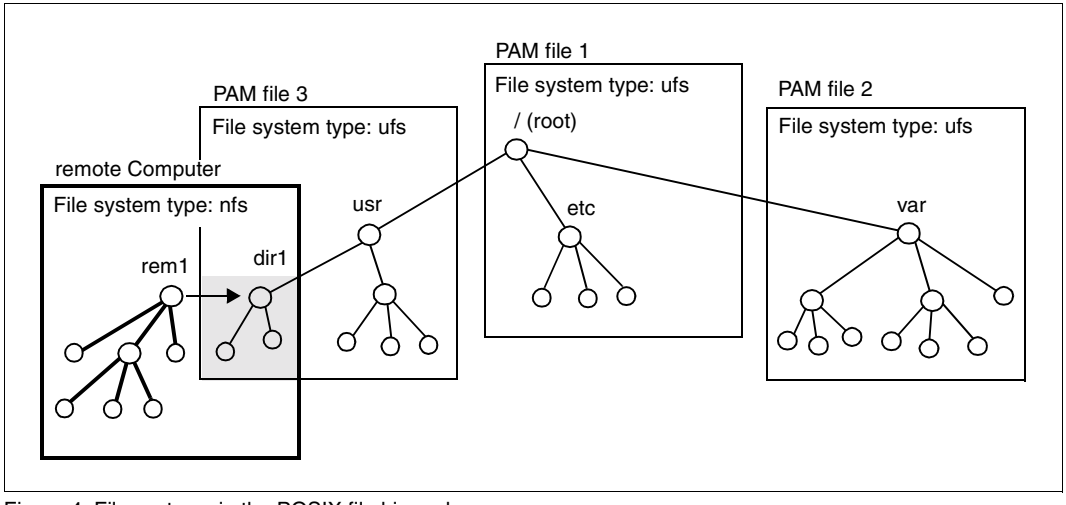


Figure 4: File systems in the POSIX file hierarchy

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 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 *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 *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. The operands -p and -V similarly provide information about mounted resources. Refer here also 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 rpcbind

– administration files

4.1 NFS commands

This section describes the NFS commands in alphabetical order. The notational conventions employed in the command syntax may be found in section “Notational conventions” on page 3.

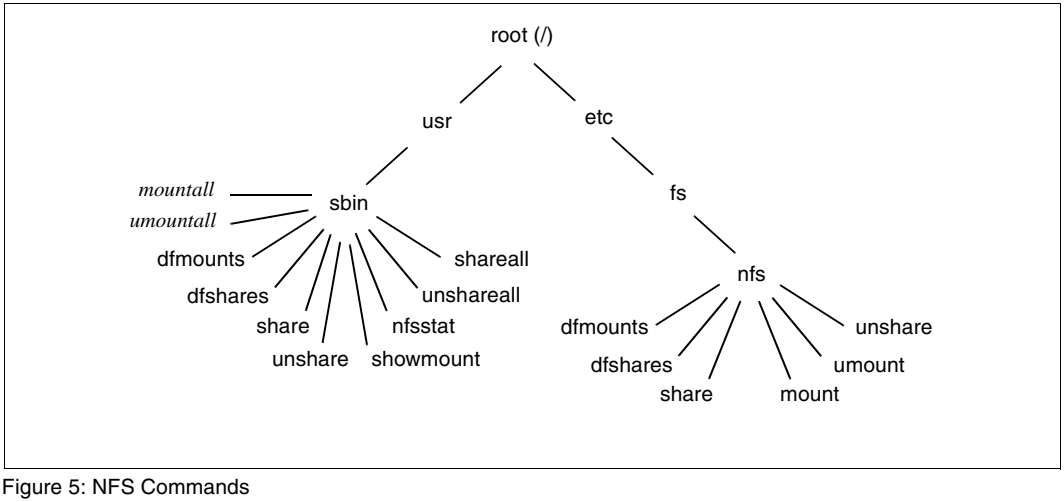


Figure 5: NFS Commands

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

Table 3: NFS commands

dfmounts Output information about mounted resources

The *dfmounts* command allows you to ascertain which resources of the local computer or of another 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 *mx4207* is providing, and on which clients they are mounted. You enter:

► **dfmounts mx4207**

The following output appears on the screen:

```
RESOURCE  SERVER  PATHNAME                CLIENTS
-         mx4207  /                       v214h429
-         mx4207  /home/pope              gabor9,pc4081,pc4083,pc4137,pc4019
-         mx4207  /home/tcp/test          v214h429
-         mx4207  /home/tcp/usr           (anon),v214h429
-         mx4207  /home2/cmx              pc4163,pc4138,pc4186,pc4185,
                                          pc4184,pc4076
-         mx4207  /home2/cmx/cmxk10/SRC-WX200/src_CMX     pc4163
-         mx4207  /home2/cmx/dos11_w      pc4163
-         mx4207  /home2/cmx/DOSF70B4     pc4076,pc4163
```

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 *lib*. The information is to be output with a header. You enter:

► `dfshares -F nfs lib`

The following output appears on the screen:

```
RESOURCE             SERVER  ACCESS          TRANSPORT
lib:/sales           lib     -               -
lib:/usr/mem         lib     -               -
lib:/usr/mod         lib     -               -
lib:/letter          lib     -               -
```

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*. *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 type *ufs*, refer to the “POSIX Commands” manual.

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.

mount 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 following 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*.

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.

secure A protocol offering a high level of security is used for NFS jobs (RPC authorization check AUTH_DES instead of the standard authorization check AUTH_UNIX). The option *secure* must be specified if the NFS server has made available the resource to be mounted with the option *secure*, see *share* command.

noac Suppresses the buffering of attributes in the buffer cache.

v2 Mounts resources of servers using NFS protocol version 2, even if the servers support protocol version 3. This option is not supported in NFS V1.2C.

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.

Buffer cache

The buffer 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 buffer 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. *actimeo=n* extends the emptying interval for normal files and for directories by *n* seconds.

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. You enter:

```
► 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 a directory with the same name *man* as the mount point. You enter:

```
► cd /home1/usr
  mkdir man
  mount -F nfs -o ro,hard,intr serv:/usr/man /home1/usr/man
```

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 that the specifications for the file systems to be mounted will come from the standard input. The individual lines are terminated with RETURN or with EM DUE. The command is executed after entry of @ @ 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

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 reinitialize the statistics counters and thus actually define the period covered by the statistics.

If the command is entered without options, all statistical information is output. No reinitialization takes place.

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 Effects the output and subsequent reinitialization (setting to zero) of the statistical information.
Which information is to be reinitialized can be specified in detail by using one of the aforementioned operands.
If only *-z* is specified, all the information is reinitialized.

RPC statistics: Server

calls Number of RPCs received.

badcalls Number of errored RPCs received (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 retransferred during a call because no acknowledgment or an errored acknowledgment was received.
badxid	Number of acknowledgments for RPC packets received after the RPC had already been concluded.
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 jobs sent.
badcalls	Number of failed NFS jobs.
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. You enter:

► `nfsstat -c`

The following output appears on the screen

```
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. You enter:

► **`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%
```

share Make local resources available for client access

The *share* command serves to make 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 suboption *ro* for individual clients.

ro=*client*[:*client*]...
Makes the resource available for read-only access to the listed clients. This specification overrides the suboption *rw* for individual clients.

anon=*uid* Assigns a user ID to unknown users.
If the AUTH_DES authorization check is used, the specified *uid* is set to the effective user ID (user number) of unauthorized users.
If the AUTH_UNIX authorization check is used, the specified *uid* is set to root.
By default, unknown users are given the effective user ID UID_NOBODY. If *uid* is set to -1, access is denied to unknown users.

root=*host*[:*host*]...
Specifies that the root users from the specified computers (*hosts*) will similarly receive root authorization on the local computer. By default, root authorization is not granted to any host.

secure A protocol offering a high level of security is used for NFS jobs (RPC authorization check AUTH_DES instead of the standard authorization check AUTH_UNIX). The option *secure* must also be specified by the clients when mounting, see *mount* command.

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 wish to make available a subdirectory of */usr/reports* called *mtgmemos* via NFS for client access. Apart from NFS, no other product for distributed file access has been installed. The directory is to be made available for all clients with read-only access. You enter:

```
▶ share -F nfs -o ro -d "MEMOS for Project X" /usr/reports/mtgnotes
```

Example 2:
You wish to make available a file system section as an NFS resource for client access; the directory *graphics* complete with all its subdirectories and files is to be made available from the file system */export*. Read-only access is to apply to all clients except the client “art.dept” which is to receive read and write authorization. You enter the following command:

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

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**␣fstype [,fstype...]] [␣– | ␣file]

-F␣fstype [,fstype...]
: Specifies that resources of the specified file system types are to be made available. If this option is not given, all resources specified in the input file or via the standard input are made available.

–
: 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 entry of @ @ 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 *misc*, via which three resources are to be made available for client access. The file contains the following entries:

```
#cat misc
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 misc`

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 the keys @ @ d. The commands are executed.
  ```

showmount
Output information about NFS clients and resources

The *showmount* command serves to provide information on the names of those client computers which have mounted resources from the local or the specified computer. The information which is outputted 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 following 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 file systems which are mounted on the client computer *hostname*. The names of the client computers are not included in the output.

-e The names of the file systems 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 *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. You enter:

► `showmount`

The following is output on your screen:

```
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 on clients. You enter:

► `showmount -d`

The following is output:

```
/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*. You enter:

► `showmount -a`

The following is output:

```
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 resources from your local computer *london* have been made available for client access using NFS and with which access authorizations they have been provided. You enter:

► `showmount -e`

The following is output:

```
export list for london:
/usr1                    (everyone)
/usr                     windsor
```

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

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*), 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 NFS type are specified in the following 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

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 of how to unmount local resources of the file system type *ufs,* 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

unshare Make resources unavailable

The *unshare* command serves to revoke 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. You enter:

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

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

unshareall Make multiple resources unavailable

The *unshareall* command serves to simultaneously revoke the availability of multiple resources. 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.

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 also lock them against access by remote NFS clients. You enter:

```
► unshareall -F nfs
```

Example 2:
You wish to make unavailable all resources currently made available by your system. You enter:

```
► unshareall
```

4.2 Daemons

This section describes the daemons in alphabetical order. The notational conventions employed in the command syntax may be found in section “Notational conventions” on page 3.

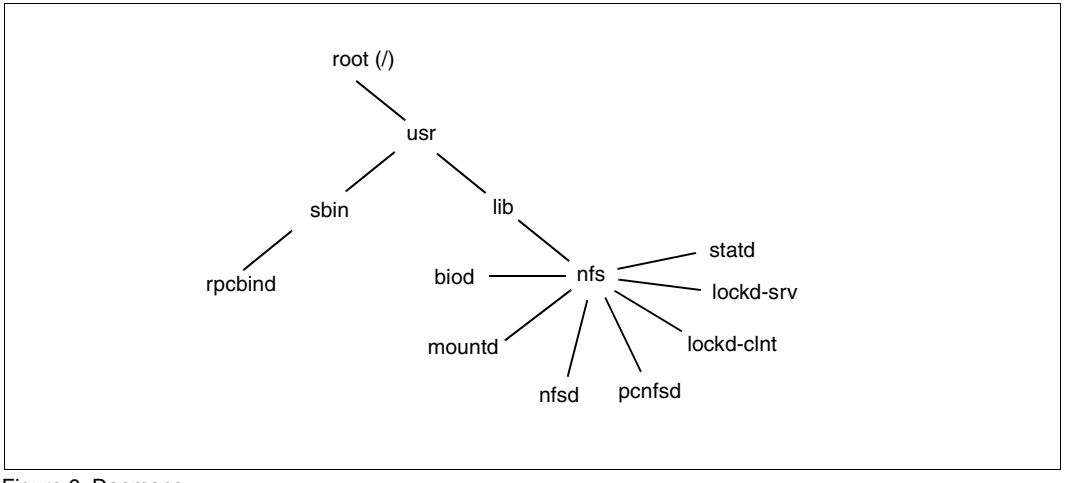


Figure 6: Daemons

The NFS daemons *biod*, *mountd*, *nfsd*, *pcnfsd, lockd-clnt, lockd-srv* 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 RPCs (remote procedure calls). 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 for dealing with the network communication which takes place via RPCs, and is supplied together with NFS in BS2000. For further information on the functions of the daemons, please refer to the publication “Managing NFS and NIS”.

The following run as standard on a BS2000 system:

```
4 nfsd
4 biod
1 mountd
1 pcnfsd
1 rpcbind
1 statd
1 lockd-srv (on NLM server and NLM client)
1 lockd-clnt (on NLM client)
```

The daemons can be monitored by entering the following POSIX command:
`ps -ef | grep` *daemonname*

The following overview lists the individual daemons:

Daemon	**Function**
biod	NFS client daemon for block-oriented input/output
lockd-clnt	Daemon for Network Lock Manager (NLM) clients
lockd-srv	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
statd	Status Monitor for status-based RPC services

Table 4: Daemons

biod NFS client daemon for block-oriented input/output

biod is a 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.

lockd-clnt Daemon for NLM clients

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

– When the client issues its first NLM request, *lockd-clnt* determines the address of the NLM server. This address is saved by the client so that it need only be queried again with *lockd-clnt* in exceptional cases.

– *lockd-clnt* registers the NLM server with the status daemon *statd*.

– *lockd-clnt* records which NLM server is accessed by the client. The list of NLM servers is periodically compared with the mount table: if an entry for an NFS is deleted from the mount table, *lockd-clnt* informs the *statd* status daemon that the associated NLM server no longer needs to be monitored.

– *lockd-clnt* establishes an RPC service that is called by *statd* when the client restarts following a crash. This server resets the client locks on the server.

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

Path: /usr/lib/nfs

Syntax

lockd-clnt

lockd-srv RPC service for NLM (Network Lock Manager)

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

One of the main tasks of *lockd-srv* is to manage blocked locks. A process attempts to set a lock, this fails, and the process blocks. If a lock that was previously blocked can be set on the server, *lockd-srv* notifies the *lockd-srv* 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 to run.

During initialization, *lockd-srv* uses *statd* to provide information to all NLM clients registered on the server. During a “locking period”, which can be defined by the system administrator, the clients can reset locks that have been lost on the server as a result of a crash, for example. Any NLM requests other than lock resets will be rejected by *lockd-srv* during this period.

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

Path: /usr/lib/nfs

Syntax

lockd-srv[␣**-g** lock_period][␣**-t**␣retry_timeout][␣**-f**␣nr_file][␣**-s**]

-g␣lock_period Locking period in seconds, during which NLM clients can reset lost locks. *lock_period* has the following format:

min**:**wait**:**max

min Minimum total time of locking period.

wait Following each lock reset, *lockd-srv* prolongs the locking period for at least another *wait* seconds, provided the maximum total time of the locking period will not be exceeded.

max Maximum total time of locking period.

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

-t␣retry_timeout

When locks block, *lockd-srv* attempts to reset 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 – normally by PC clients.

mountd Daemon for mounting remote resources

mountd is a daemon which responds to requests to mount NFS resources. To this end it reads the file */etc/dfs/sharetab*. This daemon determines which resources are available for mounting on which computers. It also provides information as to which resources have been mounted by 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

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*.

pcnfsd Daemon for supporting DOS PCs

pcnfsd is a daemon offering support for ONC clients (Open Network Computing) on PC systems under the Windows9x and Windows NT operating systems. It checks the authorizations of a PC which logs in to the NFS server with *netopen*, and supports the printout of PC files on a BS2000 printer.

The *pcnfsd* daemon services requests which are directed via RPC to program number 150001, using version 1 of the *pcnfsd* protocol.

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

Authorization: the *pcnfsd* daemon can only be started with root authorization.

Path: /usr/lib/nfs

Syntax

pcnfsd

Examples

See chapter “Connecting a Windows PC” on page 93

rpcbind Daemon for RPC

rpcbind is a daemon which converts the RPC program numbers into universal addresses. It must be running in order to be able to issue RPCs. 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. So if a client now wishes to issue an RPC 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

statd Status Monitor for status-based RPC services

statd 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 issues an RPC in order to thus utilize the service offered by the server over the network. It employs the RPC daemon to establish contact with 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.
The notational conventions employed in the command syntax may be found in section “Notational conventions” on page 3.

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 in the case of 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␣**-b**[␣**-T**␣transport]␣program␣version
rpcinfo␣**-d**[␣**-T**␣transport]␣program␣version

host	name of a remote computer. The default host is the local host. If the *rpcinfo* command is specified with the *host*, all registered RPC services are listed with *host* and *rpcbind*.
-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, except the option *-b*.

-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 use of the option *-a* is linked to the use of the option *-T*.
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.

-b
: Executes an RPC broadcast to procedure 0 of the specified *program* and the specified *version* and reports all responding hosts. If *transport* is specified, *rpcinfo* broadcasts the request only on the transport route specified through *transport*.

-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.

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 options *-b* and *-d*.

Examples

Example 1:
To show all of the RPC services registered on the local computer, use:

► `rpcinfo`

This produces the following output:

```
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  ticotsord  BS2TEST1.rpc      portmapper  superuser
 100000    3  ticots     BS2TEST1.rpc      portmapper  superuser
 100000    3  ticlts     BS2TEST1.rpc      portmapper  superuser
 100003    2  ticlts     BS2TEST1.nfsd     nfs         superuser
 100003    2     udp     0.0.0.0.8.1       nfs         superuser
 100005    1  ticlts     \007\000\000\000  mountd      superuser
 100005    1     udp     0.0.0.0.16.0      mountd      superuser
 100005    1  ticots     \003\000\000\000  mountd      superuser
 100005    1  ticotsord  \003\000\000\000  mountd      superuser
 100005    1     tcp     0.0.0.0.16.1      mountd      superuser
 150001    1     udp     0.0.0.0.2.127     pcnfsd      superuser
```

Example 2:
To show all of the RPC services registered with *rpcbind* on the computer named *klaxon,* use:

► `rpcinfo klaxon`

Example 3:
To show whether the RPC service with program number *prog_no* and version *vers* is registered on the computer named *klaxon* for the transport route *tcp*, use:

► `rpcinfo -T tcp klaxon` *prog_no vers*

Example 4:
To show all of the RPC services registered with the *rpcbind* on the local computer use:

► `rpcinfo -p`

This produces the following output:

```
program vers proto   port
 100000    3   udp    111  portmapper
 100000    2   udp    111  portmapper
 100003    2   udp   2049  nfs
 100005    1   udp   4096  mountd
 100005    1   tcp   4097  mountd
 150001    1   udp    639  pcnfsd
```

Example 5:
To check the status of version 2 of *rpcbind* (program number 100000) on the host *sparky* using the *ping* command:

► `rpcinfo -t sparky 100000 2`

Example 6:
To delete the registration for version 1 of the *walld* service (program number 100008) for all transport routes, use:

► `rpcinfo -d 100008 1`

4.4 Administration files

This section describes the administration files. The notational conventions employed in the command syntax may be found in section “Notational conventions” on page 3.

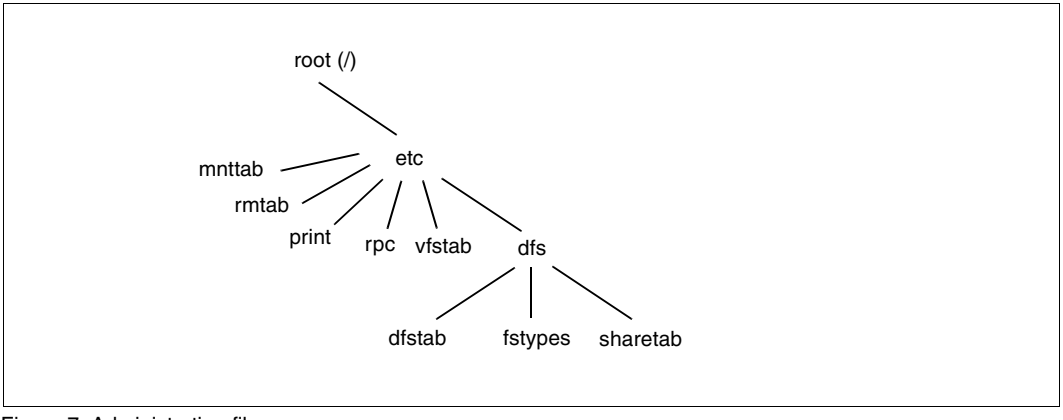


Figure 7: Administration files

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 *edt* in the POSIX shell. See the “POSIX Commands” manual.

The following overview lists the administration files:

Path	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

/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    fstype    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

Examples

In the POSIX shell enter: `cat /etc/mnttab`

```
/dev/root      /            ufs    rw,suid         802532552
/proc          /proc        proc   rw,             802532553
/dev/fd        /dev/fd      fdfs   rw              802532553
/dev/dsk/3     /var         ufs    suid,rw,noquota 802532558
/dev/dsk/2     /home1       ufs    suid,rw,noquota 802532588
SINTEST1:/nfs  /nfsclient   ufs    rw              802536261
```

/etc/print Templates for BS2000 print commands

The file */etc/print* contains templates for BS2000 print commands (refer here to the manual BS2000/OSD-BC, command PRINT-FILE).
Each line in the file comprises the following fields:

Format

```
name:code:pname:text:device:form:space:laser:chars:image:xstring:
```

name Name of the BS2000 printer as used in the *prinit* command in DOS. This name can have a maximum length of 8 characters.

code Defines whether and how the print file is to be converted to EBCDIC. To this end it is necessary to specify which conversion table is to be used. The conversion tables may be found in the directory */usr/lib/iconv*. See also the “POSIX Commands” manual, *iconv* command.

code	Table	Effect
no specification		Default (as 646)
n		No conversion is performed
646	646.edf03.t	Converts from ASCII-646 to EBCDIC-edf03; default
646da	646da.8859.t	Converts from ASCII-646da to EBCDIC-edf03;
646fr	646fr.8859.t	Converts from ASCII-646fr to EBCDIC-edf03;
646de	646de.8859.t	Converts from ASCII-646de to EBCDIC-edf03;
646it	646it.8859.t	Converts from ASCII-646it to EBCDIC-edf03;
646en	646en.8859.t	Converts from ASCII-646en to EBCDIC-edf03;
646sv	646sv.8859.t	Converts from ASCII-646sv to EBCDIC-edf03;
646es	646es.8859.t	Converts from ASCII-646es to EBCDIC-edf03;
8859	8859.edf04.t	Converts from ASCII-8859-1 to EBCDIC-edf04;

Table 6: Conversion tables

pname Name of the spoolout job in BS2000. The default is PCNFS.

text Text for the header page This name can comprise a maximum of 32 letters (D’text’). There is no default.

device Device name of the BS2000 printer. There is no default.

form Form code for laser printers. There is no default.

space Line feed control for the printer. Valid values: 1,2,3,E,A,I

- 1,2,3 Number of line feeds after each print line.
- E The file contains Siemens Nixdorf line feed characters.
- A The file contains ASA line feed characters.
- I The file contains industry-standard line feed characters.

There is no default.

laser The file is output on a laser printer (*chars* and *image* are evaluated). The default is *YES*.

chars Denotes one or more character sets which are to be used for printing the file.

image Operand for output on a laser printer. Denotes a user file which can contain LOOP, character and POOL records. If this operand is omitted, then the corresponding parameters are taken from the file *$TSOS.NDFILE*, *$TSOS.HPFILE* or *$TSOS.RSOFILE*.

xstring Extended specifications for the BS2000 PRINT-FILE command. Character string comprising a maximum of 224 characters. There is no default.

Examples

See chapter “Connecting a Windows PC” on page 93.

/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

In the POSIX shell enter: `cat /etc/rmtab`

```
SINTEST1:/nfs1/nfsserver
```

/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

In the POSIX shell enter: `cat /etc/rpc`

```
event           100101  na.event        # SunNet Manager
async-llockmgr  100020
logger          100102  na.logger       # SunNet Manager
async-nlockmgr  100021
x25.inr         100022
sync            100104  na.sync
statmon         100023
status          100024

.
.
.
ping            100115  na.ping
rpcnfs          100116  na.rpcnfs
hostif          100117  na.hostif
tfsd            100037
portmapper      100000  portmap sunrpc
etherif         100118  na.etherif
nsed            100038
rstatd          100001  rstat rup perfmeter
nsemntd         100039
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
sync_nlockmgr   200004
pcnfsd          150001  pcnfs
sync_llockmgr   200005
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
ioadmd          100055  rpc.ioadmd
```

/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*.

The file systems which are listed in the file */etc/vfstab* are mounted on starting POSIX or if the *mountall* command is entered without the option – or *file*. File systems of type *nfs* are mounted by means of *mountall -F nfs* after starting NFS.

The fields in the table are separated by blanks. A hyphen (–) indicates a blank entry in the relevant 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 block-oriented device or of the resource of the character-oriented device. When mounting remote resources, this parameter is not supported and must be replaced by the hyphen (–).

mountp — Mount point: absolute path name of the directory in which the resource is to be mounted.

fstype — File system type: *nfs* is entered for remote resources.

fsckpass — The pass number to be used for multiple *fsck* commands. When mounting remote resources, this parameter is not supported and must be replaced by the 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. The command *mountall -F nfs* is used after starting NFS to mount *nfs* file systems.

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

In the POSIX shell enter: `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
```

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

The file */etc/dfs/dfstab* contains *share* commands for making file systems 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, see page 53.

/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 system type listed in the first line of this file as the default. The file can be edited using *edt*.

Format

```
fstype designation: version
```

fstype	File system type
designation	Name of the utility package
version	Version of the utility package

Examples

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
```

/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 relevant 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

In the POSIX shell enter: `cat /etc/sharetab`

```
/nfs1/nfsserver –       nfs     rw
```

5 Connecting a Windows PC

The product NFS V3.0 enables Intel-based personal computers (PCs) with a Windows operating system to access the POSIX file system of a BS2000 computer, and permits the printing of DOS files on a printer connected to the BS2000 computer.

In contrast to UNIX implementations for Intel PCs (e.g. SCO-UNIX, Solaris, Linux), NFS is not integrated in Windows operating systems (Windows 95, Windows 98, Windows NT and Windows 2000).

To implement a Windows PC as an NFS server or client, an appropriate add-on product must be installed. Typically, a PC that mounts released server resources in the POSIX file system as a network drive is operated as an NFS client.

Fujitsu Siemens is offering the LAN1 Pro product, which is available alongside other manufacturers' implementations that contain NFS services for Windows PCs.

LAN1 Pro is based on LAN WorkPlace Pro from Novell, and comprises a suite of TCP/IP applications (Internet browser, terminal emulations, file transfer (FTP), NFS client, X-Server and email). The NFS client of LAN1 Pro enables transparent access to released POSIX directories and to printers in BS2000/OSD.

The implementation of the NFS client contained in LAN1 is described below.

5.1 Preparations

Before implementing the NFS client, you have to make some preparations in BS2000 and on the PC.

In BS2000

The POSIX user ID to which the PC connects must be assigned a standard account number for *rlogin* by means of the */MODIFY-USER-ATTRIBUTES* command.

On the PC

While you are installing LAN1 Pro using the autorun application on the CD supplied, you must manually select the services you want to install ("Custom" installation). NFS-Client is not installed with the installation variant "Typical".

Following the actual installation, you will be asked to configure the NFS client. Please note the following:

- In the "Default User-Information" dialog window displayed, you can define the default access authorization to be used when mounting NFS file systems.

  As the user name, you must specify the BS2000 ID used by the PC to establish the connection to POSIX.

  The password associated with the BS2000 ID must be entered in uppercase letters. BS2000 only accepts uppercase letters in passwords; lowercase letters are not converted automatically.

  Any logon attempt without password will be rejected by the *pcnfs* daemon as standard. However, if you want to work with an ID without a password, the password check can be disabled with the following command if SECOS is installed:

  ```
  /MODIFY_LOGON_PROTECTION ,POSIX_RLOGIN_ACCESS=*YES -
  /                        (PASSWORD-CHECK=*NO)
  ```

- Specification of NFS servers in the "Servers List" screen is optional. If you enter the names of server hosts here, these servers and their released directories will be displayed for selection in the "Map Network Drive" screen. If there is no list of server hosts, you must specify the path of the network resource (see "Processing remote file systems").

Starting NFS-Client

If you selected "Enable NFS-Client" when configuring the client, the service is activated automatically when Windows starts. Otherwise, you can start NFS-Client via the "File and Print Services" subgroup of the "LAN WorkPlace Pro" program group.

5.2 Working with remote resources

When the NFS client is started, files and directories of the POSIX file system which the NFS server makes available for remote client access, can be mounted and processed under Windows. Proceed as follows:

In POSIX (BS2000)

The file system you want to mount under Windows is made available for client access using the following command:

`share -F nfs` *resource*

resource is the full path name of the resource to be made available.

On the PC

The resource provided by the BS2000 computer is mounted under Windows in a virtual local drive.

Choose the menu command "Tools -> Map Network Drive" in Windows Explorer to mount a remote file system under a local drive letter.

The name of the NFS server and of the released file system must be specified as a path in the following format:

`\\name_of_NFS_server\file_system_path`

name_of_NFS_server:
: Name of the BS2000 computer. If the PC is connected to a DNS service, the domain name of the BS2000 host can be specified here. Otherwise, the server name must be entered in accordance with the entry in the hosts file of the PC (on WinNT: *\winnt\system32\drivers\etc\hosts*).

file_system_path:
: Absolute path of the file system released on the server (as it was specified on the server with the *share* command).

A dialog box for specifying the user name and password is then displayed (with the default values specified at installation). An authorization check by the *pcnfsd* daemon running on the BS2000 host (NFS server) is then requested, and authorization is checked using the specified BS2000 ID.

The password must be specified in uppercase letters.

5.3 Print redirection

Using NFS (BS2000) and NFS-Client on PC, it is possible to print local files from the PC on a BS2000 printer. To enable this print redirection, the *pcnfsd* daemon must have been started on the BS2000 computer (started by default by the rc script when NFS is started). In addition, there must be an entry in the administration file */etc/print* for the BS2000 printer which is to print out the PC files. Proceed as follows:

In POSIX (BS2000)

1. In the file */etc/print* of the POSIX file system enter the BS2000 printer which you wish to address from the PC.

   ```
   printer:::::::::::
   ```

   printer is the alias for the BS2000 by which name it is addressed from the PC. The fields between the colons represent parameters which can be specified for the BS2000 PRINT-FILE command. Specification of these parameters is optional.

   Example:

   ```
   BS2PRINT::DFSPRINT:‘PRINT REDIRECTION‘:(HP):STDWA4:E::(DI,DJ):::
   ```

   You have thus defined the following values for the BS20000 PRINT command:
   Name of printer: BS2PRINT
   Standard conversion
   Job name: DFSPRINT
   Text for header page: PRINT REDIRECTION
   HP printer
   Paper size: STDWA4
   SPACE=E
   Laser printer (default)
   Character sets: DI and DJ
   Standard HP file

2. Make the POSIX directory */var/spool* available for remote client access:

   ```
   share -F nfs /var/spool
   ```

On the PC

1. Configure a network printer:

   To do this, choose "Start" -> "Settings" -> "Printers" on the WinNT desktop. When you select the "Add Printer" icon in the subsequent dialog box, you can define a print server in the network with the help of the Add Printer Wizard. The path you must specify for the print server has the following syntax:

   `\\PrinterServer\PrinterName`

   PrinterServer Name of the BS2000 host.

   PrinterName Name of the BS2000 printer, as specified in the */etc/print* file.

2. Print the file:

   Using the “Print...” menu of the respective Windows application, you can select the network printer you configured in 1. above and initiate printing.

The file to be printed is placed in the POSIX directory */var/spool* as a file having the following name: `nfs`.*pcname.time*
where *pcname* is the name of the PC and *time* gives the date and time since 1.1.1970 in seconds.

The *pcnfsd* daemon on the BS2000 computer then transfers the file to the DMS (Data Management System) of BS2000 using the *bs2cp* POSIX command. It is placed under the user ID TSOS.

The *pcnfsd* daemon on the BS2000 computer fetches the template for the BS2000 PRINT command from the file */etc/print*. The file is printed via SPOOL.

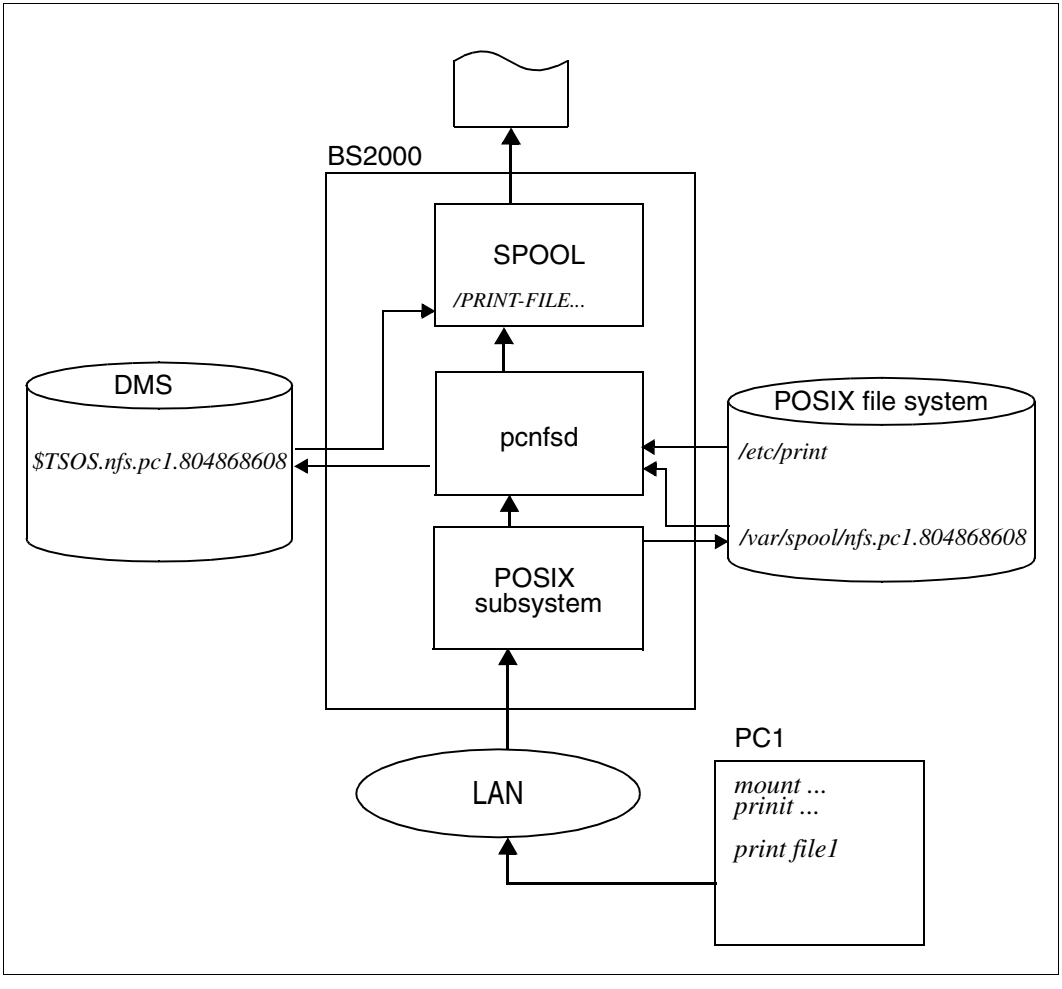


Figure 8: NFS print redirection

6 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”.

6.1 The mount operation summarized

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

Starting NFS

When *nfsstart* is entered, the 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 as to whether the *mount* command for mounting remote resources is present in the directory */etc/fs/nfs* and whether the entered command satisfies the syntax guidelines.

3. The *mount* command checks in the file */etc/mnttab* whether the resource may already have been mounted automatically.
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 made available and with which authorizations it was made available.
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. If you now wish to process a file from the mounted resource, the request is processed by the *nfsd* daemon on the computer *tanz*. Using the options specified when making the resource available, this checks whether the request can be satisfied or must be rejected.

6.2 Troubleshooting

The ability to eliminate NFS problems presupposes a thorough understanding of the circumstances under which these errors can occur. When localizing an NFS problem it is always necessary to remember that the error may have occurred in one of the three main components:
– server
– client
– network

It is therefore advisable when troubleshooting to consider these three components separately, as far as possible, in order to be able to better locate the cause of the problem.

6.2.1 Server problems

The messages which NFS outputs concerning server problems are output on the BS2000 console. They consist of a POSIX message which contains the NFS message as &00:
`POS1020  MELDUNG DES POSIX-KERNELS: &00`

The problems described in the following occur when accessing remote resources. The access can be effected by commands or from programs.
With regard to accessing resources of the server, a distinction is made between “hard” mounted and “soft” mounted directories (see *mount* command on page 43 under option *soft* and *hard*).

If a resource has been “hard” mounted and the server is unavailable for some reason or other, all programs wishing to access this resource are blocked. NFS issues the following message:

```
NFS server <hostname> not responding, still trying
```

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

```
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 hung programs to be canceled.

If a resource has been “soft” mounted and the server is unavailable, the following message is issued:

```
NFS fstat failed for server <server>: RPC: Timed out
```

In this case you should first check whether the server is active and accessible. Enter the following command, where *host* is the name of the NFS server computer (<server>) in the message:

`/usr/bin/rpcinfo` *host*

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

If the daemon is active 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`
in BS2000 command mode with: `/START-PING`

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

6.2.2 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, i.e. in the POSIX shell.

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 mounting via the file */etc/vfstab*.

```
nfs_mount: <server>:<dir>: server not responding: RPC: program not registered
```

The server which is providing the resource to be mounted is not active or the *rpcbind* daemon is not running. A further possible error cause is a faulty network connection to the server.

```
nfs_mount: <server>:/<dir>: server not responding: RPC:program unavailable
```

Although the *mount* command was able to access the *rpcbind* daemon on the server, the NFS daemon *mountd* is not registered there. A check should be carried out as to whether the NFS daemons are active on the server.

```
mount: mount-point does not exist
```

The local mount point does not exist.

`nfs_mount: access denied for <server>:/<dir>`

The resource has not been made available, 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 computer (<server>) in the message: *dfshares -F nfs server*
If the resource is not entered, 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 (<dir>) in the message: *share -F nfs pathname*

`sh: <file> cannot create`

The user does not have the requisite access rights or user number.

`mount: ...: Not a directory`

The specified path name does not refer to a directory. It may be appropriate to use the *ls* command to check whether the specified directory exists.

6.3 Performance enhancement measures

Performance problems with NFS may be attributable to various causes:

– Inadequate disk performance on the server can restrict the throughput rate at which read and write accesses can be executed.

– Excessive CPU load on client or server restricts its capability to service network requests.

– An overloaded network can restrict 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 administration 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).

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. Satellite links or other network hardware considerations may result in long delay times on the transmission path. The time limit must be increased in this case. 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.

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*, *bfs*, *rfs* and *s5*.

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. Each file is accessible via precisely one path of the file tree.

heterogeneous system environment

A computer network in which computers from different manufacturers, using different operating systems, communicate with one 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 ISO 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. In Germany, the size of the network is limited to the user's site. A LAN can be linked with other computer networks as a private subnetwork, 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. Synonym: host.

mounting

Logical (as opposed to physical) incorporation 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 establishment 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. Each layer is assigned specific functions within the framework of the overall communications task.

port number

A port number allows a particular application within a computer to be addressed. 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 group of (up to 16) shared volumes with or without system files. 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 file system at which the file tree begins. The root directory is represented by the slash (/).

RPC (remote procedure call)

The remote procedure call is the method which is used for communication between client and server in distributed processing. It is employed 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 for NFS.

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

Release (making available) of local resources for mounting on remote computers; also referred to as export.

SYSLST

BS2000 system file for print-edited lists.

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 ISO Reference Model.

transparent file access

Users can access files both on a remote computer and also on the local computer.

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 ISO Reference Model. UDP is a datagram protocol which supports broadcasting. Unlike TCP (secure end-to-end protocol), the only assurance that UDP gives is that the message has been successfully sent.

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.

Related publications

Publications for NFS

Managing NFS and NIS
Hal Stern
O'Reilly&Associates, Inc
ISBN 0-937175-75-7

Reliant UNIX V5.44
Network User's and Administrator's Guide
User Guide

Reliant UNIX V5.44
Nework Administration
System Administrator Guide

Other publications

BCAM Volume 1 (BS2000/OSD)
User Guide

BCAM Volume 2 (BS2000/OSD)
Reference Guide

BS2000/OSD-BC V4.0
Introduction to Systems Support
User Guide

BS2000/OSD-BC V4.0
Commands Volume 1 to Volume 6
User Guide

EDT (BS2000/OSD)
Statements
User Guide

PDN
User Guide

POSIX (BS2000/OSD)
Basics for Users and Administrators
User Guide

POSIX-BC (BS2000/OSD)
Commands
User Guide

SPOOL (BS2000/OSD)
User Guide

SPOOL (BS2000/OSD)
SPOOL & Print Commands
User Guide

Please apply to your local office for ordering the manuals.

Index

Contents

NFS V3.0 / NFS V1.2C (BS2000/OSD)

Network File System

Target group

NFS users and NFS administrators

Contents

The manual describes how to install and use NFS on a BS2000 computer with POSIX. It provides an overview of how to process the distributed file system NFS and a description of the commands, daemons, administration files and the trouble shooting.

Edition: February 2001

File: nfs.pdf

Fujitsu Siemens computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on NFS V3.0 / NFS V1.2C
Network File System

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.