OS/390

Network File System User's Guide

SC26-7254-00

OS/390

Network File System User's Guide

SC26-7254-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page ix.

First Edition, September 1998

This edition applies to Version 2 Release 6 of OS/390 (5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers’ comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
RCF Processing Department
M86/G26 050
5600 Cottle Rd.
SAN JOSE, CA 95193–0000
United States of America

Or you can send your comments electronically to Internet: starpubs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION"AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. DZWA

Trademarks

Summary of Changes

Summary of Changes for OS/390 Network File System User's Guide, order number SC26-7254–00, included in OS/390 Version 2 Release 6

New Information

In this release, OS/390 Network File System (OS/390 NFS) is enhanced with the following functions:

- WebNFS server and internet

  WebNFS eliminates the overhead of PORTMAP and MOUNT protocols, making the protocol easier to use through corporate firewalls. It also reduces the number of LOOKUP requests that are required to identify a particular file on the server.
- File locking over the OS/390 NFS server

  The OS/390 Network File System Network Lock Manager (OS/390 NFS NLM) and the OS/390 Network File System Network Status Monitor (OS/390 NFS NSM) are Remote Procedure Call (RPC) based servers that execute as autonomous *daemon* servers on NFS server system. NSM and NLM work together to provide file locking and access control capability with OS/390 NFS server.
- File name extension mapping support

  The OS/390 Network File System server is enhanced to provide file extension mapping for members of a Partition Data Set (PDS) and Partition Data Set Extended (PDSE) that are mounted via the Network File System from a client machine. The file name extension mapping capability is provided by the use of *side files* on the MVS host specified by the system administrator and the user during the MOUNT operation.
- Sun NFS Version 3 protocol support for both client and server functions

  The NFS version 3 protocol is a revision of the NFS version 2 protocol. It supports larger files and file systems by allowing 64-bit sizes and offsets. The NFS version 3 protocol enhances performance by returning attributes for every procedure, thus reducing the number of calls requesting modified file attributes. The NFS version 3 also enhances performance by adding support to allow the NFS server to do asynchronous writes and by relaxing the limitations on transfer sizes.
- TCP support

  In addition to User Datagram Protocol (UDP), OS/390 NFS server supports connection-oriented reliable TCP. Any client platform with TCP support can choose to access the OS/390 NFS server using TCP.

Changed Information

- Added text in support of the OS/390 NFS client enhancements.
- Added text in support of the OS/390 NFS server enhancements.
- Added and revised the OS/390 NFS messages.
- Added new terms to Glossary.
- The title of this publication has changed from *DFSMS/MVS Network File System User's Guide* to *OS/390 Network File System User's Guide.*
- As part of the name change of OpenEdition to OS/390 UNIX System Services, occurrences of OS/390 OpenEdition have been changed to OS/390 UNIX System Services or its abbreviated name, OS/390 UNIX. OpenEdition can continue to appear in messages, panel text, and other code with OS/390 UNIX System Services.

This publication is a revision in support of the functional changes introduced with OS/390 NFS. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change. For a book that has been updated in softcopy only, the vertical lines indicate changes made since the last printed version.

This revision also includes maintenance and editorial changes.

Preface

This preface contains information about this book, required product knowledge, information on how to tell if a book is current, a table of related publications, a table of referenced publications, information on how to read syntax diagrams, and information on conventions used in this book.

About This Book

This book provides the OS/390 NFS client user and the administrator with information about using the OS/390 NFS, including:

- An introduction to the OS/390 NFS.
- A description and examples of the commands used to access data sets on the MVS host processor, and a description and examples of the commands used to display and override the data set (file) attributes.
- A description of how to create your own data sets on the host processor, with descriptions (and examples) of the required attributes.
- A description and examples of the OS/390 NFS client commands used to access data on systems which support the NFS protocols. The remote NFS server can be an MVS, UNIX**, AIX, OS/2 or other system.
- A description of how the OS/390 NFS Network Lock Manager and the OS/390 NFS Network Status Monitor run with OS/390 NFS.
- Explanations of the messages sent to the OS/390 NFS client.
- A list of related protocol specifications.
- Information on handling the file size value.
- Information on handling the time stamps.
- A list of acronyms and definitions of terms used in this book.

In this book, the OS/390 NFS server is also referred to as *the server*.

Required Product Knowledge

If you are creating or accessing conventional MVS data sets, you should be familiar with the MVS file system. You should also be familiar with the operating environment of the system that you are using to run the Network File System client software.

How to Tell if this Book is Current

IBM regularly updates its books with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a book are identical, but subsequent updates might be available in softcopy before they are available in hardcopy. Here's how to determine the level of a book:

- Check the book's order number suffix (often referred to as the dash level). A book with a higher dash level is more current than one with a lower dash level. For example, in the publication order number SC26-4930-02, the dash level 02 means that the book is more current than previous levels, such as 01 or 00. Suffix numbers are updated as a product moves from release to release, as well as for hardcopy updates within a given release.

- Check to see if you are using the latest softcopy version. To do this, compare the last two characters of the book's file name (also called the book name). The higher the number, the more recent the book. For example, DGT1U302 is more recent than DGT1U301.
- Compare the dates of the hardcopy and softcopy versions of the books. Even if the hardcopy and softcopy versions of the book have the same dash level, the softcopy could be more current. This will not be apparent from looking at the edition notice. The edition notice number and date remain that of the last hardcopy version. When you are looking at the softcopy product bookshelf, check the date shown to the right of the book title. This will be the date that the softcopy version was created.

  Also, an asterisk (*) is added next to the new and changed book titles in the CD-ROM booklet and the README files.

Vertical lines to the left of the text indicate changes or additions to the text and illustrations. For a book that has been updated in softcopy only, the vertical lines indicate changes made since the last printed version.

Where to Find More Information

Where necessary, this book references information in other books, using the shortened version of the book title. For complete titles and order numbers of the books for all products that are part of OS/390, see *OS/390 Information Roadmap* GC28-1727.

Related Publications

For information about TCP/IP, refer to:

Publication Title	Order Number
AIX Operating System TCP/IP User's Guide	SC23-2309
TCP/IP for MVS: User's Guide	SC31-7136
TCP/IP for MVS: Customization and Administration Guide	SC31-7134
TCP/IP for MVS: Programmer's Reference	SC31-7135

For information about OpenEdition MVS, refer to:

Publication Title	Order Number
OS/390 UNIX System Services Programming Tools	SC28-1904
OS/390 UNIX System Services Command Reference	SC28-1892
OS/390 UNIX System Services User's Guide	SC28-1891

For information about the IBM Time Sharing Option (TSO), refer to:

Publication Title	Order Number
OS/390 TSO/E User's Guide	SC28-1968

For information about SMP/E, refer to:

Publication Title	Order Number
OS/390 SMP/E User's Guide	SC28-1740

For information about RS/6000 communications, refer to:

Publication Title	Order Number
AIX Commands Reference for RISC System/6000, Volume 1	GC23-2376
AIX Commands Reference for RISC System/6000, Volume 2	GC23-2366
AIX Commands Reference for RISC System/6000, Volume 3	GC23-2367
AIX Commands Reference for RISC System/6000, Volume 4	GC23-2393
AIX Communications Concepts and Procedures for IBM RISC System/6000	GC23-2203

Referenced Publications

Within the text, references are made to these publications:

Short Title	Publication Title	Order Number
Character Data Representation Architecture Reference and Registry	*Character Data Representation Architecture Reference and Registry*	SC09-2190
Character Data Representation Architecture Overview	*Character Data Representation Architecture Overview*	GC09-2207
DFSMS/MVS DFSMSdfp Storage Administration Reference	*DFSMS/MVS DFSMSdfp Storage Administration Reference*	SC26-4920
DFSMS/MVS V1R4 Online Product Library	*DFSMS/MVS Version 1 Release 4 Online Product Library*	LK2T-8731
DFSMS/MVS Using Data Sets	*DFSMS/MVS Using Data Sets*	SC26-4922
IBM Online Library: AIX Base Collection Kit	*IBM Online Library: AIX Base Collection Kit*	SK2T-2166
IBM Online Library Omnibus Edition: MVS Collection	*IBM Online Library Omnibus Edition: MVS Collection*	SK2T-0710
IBM Online Library Omnibus Edition: OS/2 Collection	*IBM Online Library Omnibus Edition: OS/2 Collection*	SK2T-2176
IBM Networking Softcopy Collection Kit	*IBM Networking Softcopy Collection Kit*	SK2T-6012
OS/390 MVS JCL Reference	*OS/390 MVS JCL Reference*	GC28-1757
OS/390 Network File System Customization and Operation	*OS/390 Network File System Customization and Operation*	SC26-7253
OS/390 Version 2 Release 6 Network File System User's Guide	*OS/390 Network File System User's Guide*	SC26-7254
OS/390 UNIX System Services Planning	*OS/390 UNIX System Services Planning*	SC28-1890
OS/390 UNIX System Services Command Reference	*OS/390 UNIX System Services Command Reference*	SC28-1892
OS/390 UNIX System Services File System Interface Reference	*OS/390 UNIX System Services File System Interface Reference*	SC28-1909

Short Title	Publication Title	Order Number
OS/390 UNIX System Services Messages and Codes	*OS/390 UNIX System Services Messages and Codes*	SC28-1908
OS/390 Security Server (RACF) General User's Guide	*OS/390 Security Server (RACF) General User's Guide*	SC28-1917
TCP/IP for MVS: Customization and Administration Guide	*TCP/IP for MVS: Customization and Administration Guide*	SC31-7134

How to Read Syntax Diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. Keyword parameters are parameters that follow the positional parameters. Unless otherwise stated, keyword parameters can be coded in any order. The following list tells you how to interpret the syntax diagrams:

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with two arrowheads facing each other.

```
►►──┤ Syntax Diagram ├──────────────────────────────────────────────►◄
```

- If a diagram is longer than one line, each line to be continued ends with a single arrowhead and the next line begins with a single arrowhead.

```
►►──┤ First Line ├──────────────────────────────────────────────────►◄


►►──┤ Second Line ├─────────────────────────────────────────────────►◄


►►──┤ Last Line ├───────────────────────────────────────────────────►◄
```

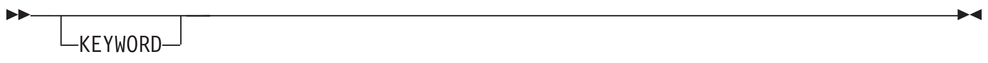
- Required keywords and values appear on the main path line. You must code required keywords and values.

```
►►──REQUIRED_KEYWORD────────────────────────────────────────────────►◄
```

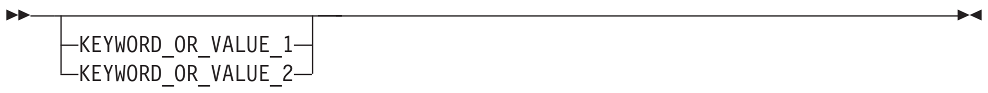
  If several mutually exclusive required keywords or values exist, they are stacked vertically in alphanumeric order.

```
►►──┬─REQUIRED_KEYWORD_OR_VALUE_1─┬─────────────────────────────────►◄
    └─REQUIRED_KEYWORD_OR_VALUE_2─┘
```

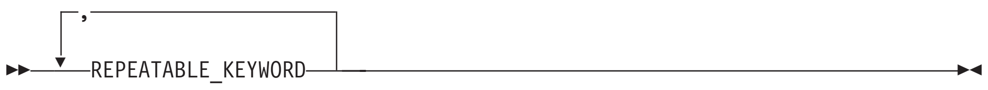
- Optional keywords and values appear below the main path line. You can choose not to code optional keywords and values.



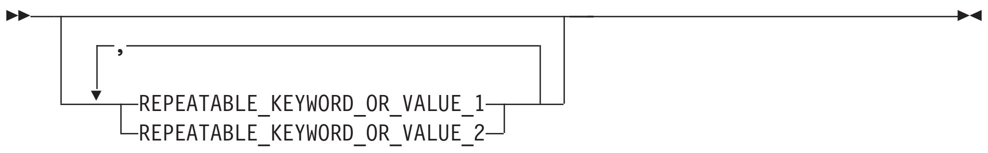
If several mutually exclusive optional keywords or values exist, they are stacked vertically in alphanumeric order below the main path line.



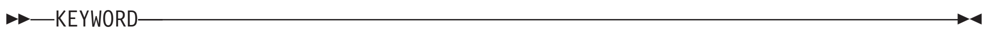
- An arrow returning to the left above a keyword or value on the main path line means that the keyword or value can be repeated. The comma means that each keyword or value must be separated from the next by a comma.



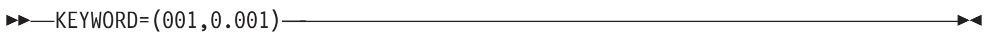
- An arrow returning to the left above a group of keywords or values means more than one can be selected, or a single one can be repeated.



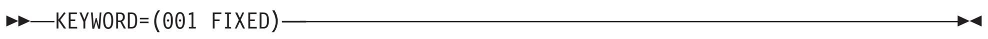
- A word in all uppercase is a keyword or value you must spell exactly as shown. In this example, you must code ***KEYWORD***.



  If a keyword or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***KEYWORD=(001,0.001)***.



- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***KEYWORD=(001 FIXED)***.



- Default keywords and values appear above the main path line. If you omit the keyword or value entirely, the default is used.

```
      ┌─DEFAULT─┐
►►────┴─KEYWORD─┴──────────────────────────────────────────────────────►◄
```

- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

```
►►──variable───────────────────────────────────────────────────────────►◄
```

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

```
              (1)
►►──KEYWORD──────────────────────────────────────────────────────────►◄
```

Notes:

1 An example of a syntax note.

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

```
►►──┤ Reference to Syntax Fragment ├───────────────────────────────────►◄
```

Syntax Fragment:

```
├──1ST_KEYWORD,2ND_KEYWORD,3RD_KEYWORD──────────────────────────────────┤
```

Conventions Used in this Book

The conventions used in this book are defined below.

, Commas are used as separators when specifying data set attributes.

( ) Parentheses in data set attributes must be passed to MVS, but the AIX or UNIX client systems' shells recognize parentheses and strip them off. To prevent this, use one of these methods:

- Include the attributes in double (") or single (') quotation marks
- Include each parenthetical expression in quotation marks
- Precede each parenthesis with a back slash (\ ).

$ The dollar sign can be used in the file name of an MVS mounted data set on UNIX. To prevent UNIX from trying to interpret $ as part of a variable, use one of these methods:

- Include the entire file name in single (') quotation marks (for example, `vi '$file2'`)
- Precede each file name with a back slash (\ ) (for example, `vi \$file2`).

	The dollar sign is also used as a prompt in the screen examples indicating that any user can enter the command.
#	Character in an AIX or UNIX prompt indicating that superuser authority is required to enter the command.

Prompts for users and superusers vary depending on the original AIX or UNIX implementation, or on site defaults. This book follows the $ and # conventions outlined above for illustration.

This book uses a “railroad track” syntax convention to show how to enter commands. To clarify the explanations, both railroad track diagrams and screen samples are used when illustrating commands.

Commands for AIX and UNIX implementations are case sensitive. Network File System commands issued at the client are shown in lowercase letters.

Chapter 1. Introduction to the Network File System

This chapter explains the NFS client-server relationship and introduces the IBM Network File System (OS/390 NFS). Read this chapter to learn about the Network File System. When used to access OS/390 UNIX System Services (OS/390 UNIX) data, which conforms to Portable Operating System Interface (POSIX**) standards, it is similar to other UNIX/AIX NFS** systems.

Overview

A *client* is a computer or process that requests services on the network. A *server* is a computer or process that responds to a request for service from a client. A *user* accesses a service, which allows the use of data or other resources.

Figure 1 illustrates the client-server relationship. The upper right portion of the figure shows the OS/390 NFS server. The lower right portion of the figure shows the OS/390 NFS client. The left portion of the figure shows various NFS clients and servers which can interact with the OS/390 NFS server and client. The center of the figure shows the Transmission Control Protocol/Internet Protocol (TCP/IP) network used to communicate between the clients and servers.

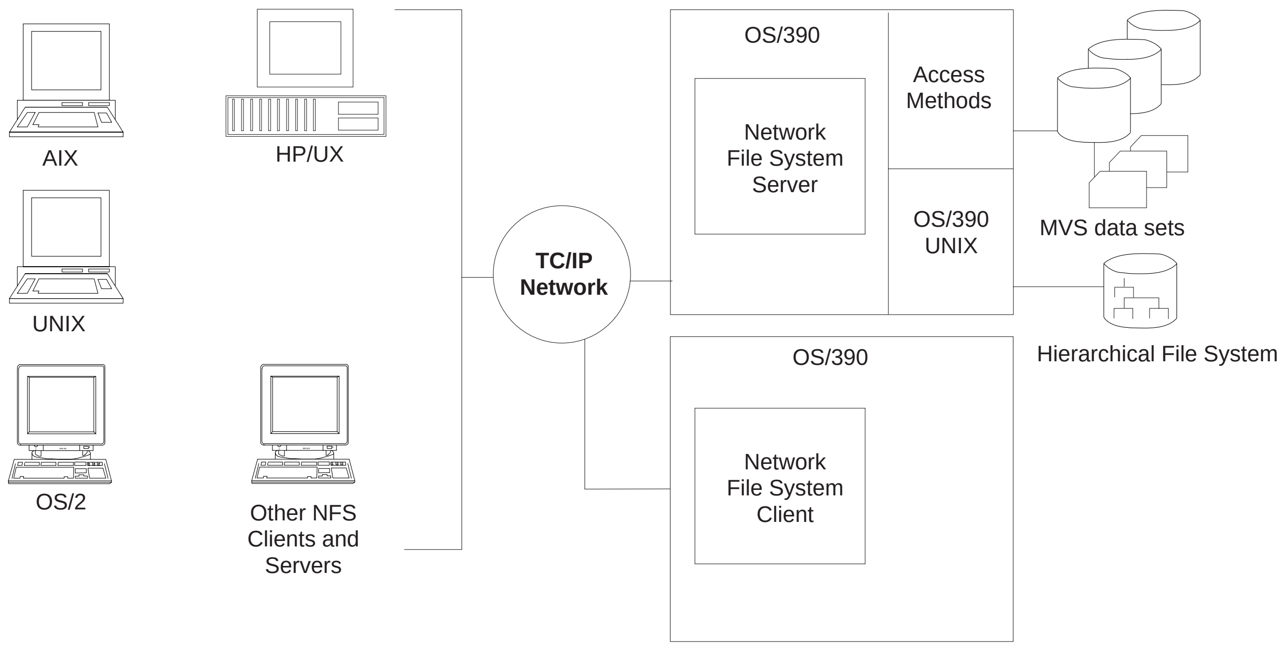


Figure 1. Network File System Client-Server Relationship

With the OS/390 NFS server, you can remotely access MVS/ESA conventional data sets or OS/390 UNIX files from workstations, personal computers, and other systems that run client software for the Sun NFS version 2 protocols, the Sun NFS version 3 protocols, and the WebNFS protocols over TCP/IP network.

The OS/390 NFS server acts as an intermediary to read, write, create or delete OS/390 UNIX files and MVS data sets that are maintained on an MVS host system. The remote MVS data sets or OS/390 UNIX files are mounted from the host processor to appear as local directories and files on the client system. This server makes the strengths of an MVS host processor — storage management, high-performance disk storage, security, and centralized data — available to the client platforms.

With the OS/390 NFS client you can allow basic sequential access method (BSAM), queued sequential access method (QSAM), virtual storage access method (VSAM), and OS/390 UNIX users and applications transparent access to data on systems which support the Sun NFS Version 2 protocols and the Sun NFS Version 3 protocols. The remote NFS Server can be an MVS, UNIX**, AIX, OS2, or other system. The OS/390 NFS client is implemented on OS/390 UNIX and implements the client portion of the Sun NFS Version 2 protocols and the Sun NFS Version 3 Protocols.

The Network File System uses the communication services provided by TCP/IP, a suite of protocols that includes the Remote Procedure Call (RPC) and External Data Representation (XDR) protocols. RPC allows a program on one machine to start a procedure on another machine, as if the procedure is local. XDR resolves the differences in data representation of different machines.

The Network File System, then, can be used for:
- File sharing between platforms
- File serving (as a data repository)

If you are using Network File System as a file server, the Hierarchical File System might be a better choice than using conventional MVS data sets, because of its UNIX-like features.

Using OS/390 UNIX Files

The Network File System server enables the client user remote access to OS/390 UNIX files from a client workstation.

OS/390 UNIX provides a Hierarchical File System (HFS) for MVS. The HFS file system is similar to a UNIX** file system. All OS/390 UNIX files reside in a directory, which in turn is a file in a higher level directory. The highest level directory is called the *root directory*.

When client users **mount** files from your server system, you use a common HFS prefix to distinguish OS/390 UNIX files from conventional MVS data sets. You see OS/390 UNIX files in a standard UNIX format on your workstation, but the files are stored on an MVS host system.

Using the Network File System, the client can **mount** all or part of the OS/390 UNIX file system and make it appear as part of your local file system. From there the client user can create, delete, read, write, and treat the host-located files as part of the workstation's own file system. For more information on OS/390 UNIX, refer to *OS/390 UNIX System Services User's Guide*.

OS/390 UNIX Enhancements

OS/390 UNIX file system support provides these enhancements over conventional MVS data sets:
- Support for hierarchical directories
- File names up to 255 characters in length
- Path names up to 1023 characters in length

- Mixed case names and special characters, except nulls, slash, and comma characters, are permitted in file and path names
- UNIX-style access permission support
- Group and user ID support at a file level
- Ability to link conventional MVS data sets to a POSIX path name

NFS Protocol Compliance

The Network File System provides full NFS protocol compliance for accessing the HFS file system.

Using Conventional MVS Data Sets

Using the Network File System, you can access conventional MVS data sets from a client workstation, personal computer, or any client system using software for the NFS protocol.

In MVS, a file is called a *data set*. The Network File System allows client users to mount conventional MVS data sets from their workstations. It presents the information to them in the form of a UNIX (or AIX), OS/2, or DOS file, though the information is actually stored on an MVS-owned DASD.

The files for an operating system are organized into a *file system*. The UNIX, OS/2, and DOS environments use a file system which is a hierarchy of *directories*. Conventional MVS, in contrast to OS/390 UNIX, uses a non-hierarchical file system in which groups of data sets are referred to by specifying a *high-level qualifier* (HLQ).

The MVS HLQ can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the HLQ for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the HLQ of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

Mounting MVS Data Sets onto a Client Mount Point

To access an MVS file system from your client, client users use the **mount** command to create a temporary link (until unmounted) between specific MVS data sets and your UNIX directory (preferably empty) or an unused logical drive on their workstations. The empty UNIX directory or logical drive is called a *mount point*.

Client users use an MVS HLQ in the **mount** command to specify which MVS data sets to **mount** at a *mount point*. The MVS data sets beginning with the specified HLQ appears as files under the *mount point*.

Client users can also perform a **mount** using a fully qualified data set name or an alias to a user catalog, but not the catalog name itself. Only Integrated Catalog Facility (ICF) cataloged data sets are supported by the OS/390 NFS server. Tape data sets and generation data sets are not supported.

Some client platforms support TCP in addition to UDP. Users can choose either TCP or UDP to access the server. The default protocol option depends on the NFS client platform.

Note: When directly mounting on a fully qualified data set name, the server must return the mount size as part of getting the attributes for the mount. This can slow down the completion of the mount command.

Creating Conventional MVS Data Sets

Client users can create MVS data sets from a client system using the Network File System. The default *data set creation attributes* specified by the system administrator are used to create MVS data sets, unless the user overrides them. These attributes determine how the MVS data sets are structured and where they are stored. Client users can override the default data set creation and processing attributes for a *mount point* when issuing the **mount** command. In addition, you can override these attributes at file creation time.

Data Set Serialization and Sharing

The OS/390 NFS server handles data set serialization and sharing differently depending on the type of data set:

physical sequential
The server insures the physical sequential data set read/write integrity by SVC 99 dynamic allocation with exclusive option whenever a physical sequential data set is opened for output. Otherwise, it allocates with share option.

VSAM The server dynamically allocates a VSAM data set with share option and allows the VSAM access method to manage data sharing using the SHAREOPTIONS specified during data set definition.

PDSE The server dynamically allocates a PDSE data set with share option and allows the PDSE functions to manage the serialization of the PDSE data set and its members.

PDS For read and write, the OS/390 NFS server issues ENQ SHR on QNAME=SYSDSN and RNAME=dataset_name (through an SVC 99). For write, the server issues an exclusive ENQ against QNAME=SPFEDIT and RNAME=dataset_name.member_name, in addition to the serialization of resources by SVC 99. For all MVS users who are allocating their data set with exclusive status, this provides write protection. It only provides read integrity for ISPF users.

OS/390 NFS Server Control Files

These special files are used by the MVS system administrator to control the OS/390 NFS server:
- Attributes data set
- Exports data set
- Mount handle data sets
- Log data sets
- Checklist data set

The Attributes Data Set

The *attributes data set* contains the settings for the OS/390 NFS server. There are three types of attributes stored in this data set:

Data set creation attributes
Used to define the structure of MVS data sets when creating a file (for conventional MVS data sets only).

File processing attributes
Used to control how files are accessed by the client.

Site attributes
Used to control OS/390 NFS server resources.

The system administrator changes the default settings by editing the attributes data set and restarting the server. Client users can override the data set creation and file processing attributes at the command line. For conventional MVS data sets, the client user can specify the data set creation attributes when mounting, creating, or accessing files. The client user can override the file processing attributes when mounting, creating, or accessing files. However, some file processing attributes can only be overridden on a *mount point* basis.

Note: Many of the attributes are valid for conventional MVS files only, and not for OS/390 UNIX files. "Attributes Used for OS/390 UNIX System Services File Access" on page 89 gives a complete list of attributes that are valid for OS/390 UNIX.

The Exports Data Set

The *exports data set* can control which client users can **mount** which MVS data sets. The entries in the exports data set specify which MVS high-level qualifiers or HFS directories can be mounted. The system administrator can use this data set to limit mounts to accredited clients only. It also controls which client users can **mount** all or part of the OS/390 UNIX file system, based on the client machine's specified Internet Protocol (IP) address. To use the exports data set, the **security** site attribute must be set to either **safexp** or **exports** by the MVS system administrator.

The Mount Handle Data Sets

The OS/390 NFS server maintains a list of the active *mount points* in a pair of files called the mount handle data sets on MVS. The two data sets are used alternately to automatically reestablish the client *mount points* when the server is started. If the file system is not available, the mount point is not reestablished and the mount failure is recorded in the LOG data set.

The OS/390 NFS server does the cleanup activity during OS/390 NFS server shutdown and daily at the cleanup time specified by the **restimeout** site attribute.

During cleanup time, the OS/390 NFS server reads the list and checks all mount points against the retention period specified in the **restimeout** site attribute. If your *mount points* are idle longer than the retention period specified in the **restimeout** site attribute, they are removed. Only the active *mount points* are reconnected.

If a mount handle is removed by the cleanup activity, the client user might receive the "Stale NFS File Handle" message or some other appropriate message. If so, all the client user needs to do is **unmount** the stale *mount point* and **mount** it again.

The Log Data Sets

The *log data sets* store the messages for the OS/390 NFS start-up procedures. This log can be used to identify the users correctable errors or the users problem errors. There are two logs that this information is stored in; the primary log and the secondary log. The primary log is used at start-up until it is filled and then overflows into the secondary log. When the secondary log is full, the primary log will then be overwritten with new error messages. The number of log records is dependent on the number of transactions that the server can handle.

The Checklist Data Set

The checklist data set contains entries for files or directories that are to be exempt from security authorization facility (SAF) checking even though SAF or SAFEXP is specified as the security options. This file is only used when SAF or SAFEXP is specified for the particular data type and the CHECKLIST option is specified as a site attribute. The entries specified here must match a subsequent mount point, or be the parent of a subsequent mount point, to allow SAF checking to be bypassed for everything underneath that mount point. If the entry does not match a subsequent mount point and it is not a parent of any subsequent mount point then it has no effect.

Supported Clients for the OS/390 NFS Server

IBM Systems Available for Network File System Client use

- OS/390 NFS
- IBM RS/6000 AIX Version 4.2.0

Network File System client software for other IBM platforms is available from other vendors. You can also access the OS/390 NFS server from non-IBM clients that use the NFS version 2 protocol including:

- DECstations** running DEC ULTRIX** version 4.4
- HP 9000 workstations running HP/UX** version 10.20
- Sun PC-NFS** version 5

You can also access the OS/390 NFS server from non-IBM clients that use the NFS version 3 protocol including:

- Sun** workstations running SunOS** or Sun Solaris** versions 2.5.3.
- AIX 4.2.0

Note: This information reflects supported clients as of September 1998. For current information on supported clients, please contact IBM.

Supported Servers for the OS/390 NFS Client

The OS/390 NFS client supports all servers which implements the server portion of the Sun NFS Version 2 or Version 3 protocols.

A new MOUNT parameter **vers(x)**, where *x* is either 2 or 3, is provided to make the OS/390 NFS client to communicate with the server at the protocol level specified. The OS/390 NFS client also communicates at the highest protocol level supported by the server if no level is specified.

If no version is specified:

- If the server only supports NFS version 2 protocol then the OS/390 NFS client will use NFS version 2 protocol to communicate.
- If a server supports both the NFS version 2 and 3 protocols, then OS/390 NFS client will use NFS version 3 protocol to communicate.

vers(2)
Use NFS version 2 protocol to communicate with the server.

vers(3)
Use NFS version 3 protocol to communicate with the server. OS/390 NFS client fails the MOUNT command if the server does not support NFS version 3 protocol.

WebNFS Support

The OS/390 Network File System server supports the WebNFS protocol. WebNFS specification extends the semantics of NFS versions 2 and 3 protocols to allow clients to obtain file handles without the MOUNT protocols. The OS/390 NFS server supports the public filehandle and multi-component lookup features as well as other additional requirements as described in RFC 2055. A new keyword, **public**, is added for the system administrator to specify the public paths that the public file handle can access. A public path for conventional MVS data and a public path for HFS data can both be specified. When a LOOKUP request comes in from an NFS client and an absolute path name is specified, it will be matched with the public paths to determine which public path it is trying to reference. If a relative path is specified and both HFS and MVS public paths are defined then the LOOKUP request will be processed relative to the HFS public path.

The following are restrictions for the WebNFS support provided by the OS/390 NFS server in this release.

- Export Spanning Pathnames

  LOOKUP requests, which reference files or directories outside of the exported public path, will result in an error condition.
- Symbolic Links

  A symbolic link embedded in a multi-component pathname LOOKUP request will result in an error condition. However, if the final component is a symbolic link, the server will return the filehandle of the symbolic link and let the client evaluate it. External links, which are special cases of symbolic links, will be handled similarly.
- Native Path

  Only canonical pathnames will be supported.

The NFS Version 3 and TCP Protocol

The information for NFS version 3 protocol and **proto=tcp** can be found on the *mount* man page on a UNIX client. The NFS client automatically selects the option unless the end-user overrides the option. For example:

```
unix$ mount -o vers=2,proto=udp mvshost1:smith /mnt
```

The above example shows the preference of NFS version 2 with **udp** protocol, even though the client platform can handle the NFS version 3 and **tcp** protocol.

Users can issue the `rpcinfo -p <hostname>` to show all the RPC programs available on the server. Table 1 illustrates this example.

$ rpcinfo -p mvshost1

Table 1. View of NFS Server Capability

program	vers	proto	port	service
100000	2	udp	111	portmapper
100000	2	tcp	111	portmapper
100003	2	udp	2049	nfs
100003	**3**	**udp**	**2049**	**nfs**
100059	2	udp	2012	
100044	1	udp	2013	
100005	1	udp	2014	mountd

Table 1. View of NFS Server Capability (continued)

program	**vers**	**proto**	**port**	**service**
100005	3	udp	2015	mountd
150001	1	udp	2016	pcnfsd
150001	2	udp	2017	pcnfsd
100044	**1**	**tcp**	**2010**	
100059	**2**	**tcp**	**2011**	
100003	**2**	**tcp**	**2049**	**nfs**
100003	**3**	**tcp**	**2049**	**nfs**

Note: New information is represented in **bold**.

Chapter 2. Using Conventional MVS Data Sets

This chapter explains what you need to know to use conventional MVS data sets on a client workstation. This chapter explains:

- Special MVS considerations
- Reading and writing MVS data sets
- Accessing MVS data sets
- Mapping between the workstation and MVS file systems

Special MVS Considerations

In addition to mapping between the workstation and MVS file systems, there are other ways in which the OS/390 NFS server might differ from non-MVS Network File System servers. These differences include:

- Selecting an MVS data storage format
- File size determination and time stamps
- Ownership and permissions
- Statelessness
- Reading and Writing Files
  - Random Access to Files
  - Cached data writing
- Case Sensitivity — maplower, nomaplower
- Selecting Text or Binary Processing Modes — text, binary
  - Binary Processing Mode
  - Text Processing Mode
- Number Representation
- Accessing Migrated Files — retrieve, noretrieve; wait, nowait
- Obtaining Attributes of Migrated Data Sets Without Recall
- File Handle refresh
- File Extension Mapping

Selecting an MVS Data Storage Format

The files you create with the OS/390 NFS server are contained in MVS data sets. These MVS data sets are record-oriented and can be sequential, direct, VSAM, partitioned, and so forth. These MVS data sets are variable or fixed in record length. UNIX, OS/2, and DOS files, however, are byte-oriented and typically written or read at certain offsets in the file.

You can map non-MVS files to most types of MVS data set organizations. However, how the time stamps and file size value are handled depends on the type of MVS data set used, and the file size processing can affect performance. Refer to “Appendix C. Handling of the File Size Value” on page 115.

Direct reads with record format recfm(fbs) or recfm(f) can be fast. In some cases, the OS/390 NFS server can determine the physical block addresses from the record offsets. The MVS sequential file organization with recfm(fbs) or recfm(f) on DASD allows for efficient updating or reading at any offset in the file. Other supported MVS access methods (for example, VSAM) can be used if required by a given application but, in general, the sequential file organization is the best choice for files that are used mainly by UNIX clients.

File Size Determination and Time Stamps

How the OS/390 NFS server handles the file size value and time stamps depends on the type of MVS data set used and the attributes used to access the data set. Refer to "Appendix C. Handling of the File Size Value" on page 115 and "Appendix D. Handling of the Time Stamps" on page 119.

Ownership and Permissions

The UNIX `UID` and `GID` file attributes are reset to their default state after a restart of the OS/390 NFS server or an unmount of the file system. In some cases, this requires that a superuser on the client workstation reissue a `chmod` command to reset the `UID` and `GID`. This command can be included in the same script used to mount the file system.

The permissions checking done by RACF, or an equivalent security package, is transparent to you. Access to a data set is granted, provided that the server's exports list, the MVS security subsystem, and the customized installation security exit allow access to the data set. Which of these security systems are active depends on the security settings used at your installation. The UNIX file modes or permission bits are ignored by OS/390 NFS server and authorization is done with the RACF or equivalent security package.

Statelessness

The OS/390 NFS server strives to be as stateless as possible; that is, it tries to work correctly without maintaining any state information about any of its clients. However, a failure of the server causes cached writes to be lost, some attributes to be reset, and file handles to become stale, or not valid.

Reading and Writing Files

After the OS/390 NFS server is started and you have mounted the MVS data set, you can use regular data access or creation commands from your workstation to access files that reside on MVS.

For example, suppose you accessed an MVS file named *prefix.*`file3` mounted on the local directory */mnt*. This is how you could use the UNIX **cat** command (or a similar command) to display the file:

```
$ cat /mnt/file3
```

Suppose you accessed an MVS file named *prefix.*`file12` mounted on the local directory */mnt*. This is how you could use the UNIX **vi** command (or a similar command) to edit the file:

```
$ vi /mnt/file12
```

Writing a file on MVS is straightforward. If the file already exists, the file's existing attributes are used; they are not modified during the write operation. For the priorities of attributes, see "Overriding Data Set Creation Attributes" on page 21.

Random Access to Files

If your application accesses the files at random offsets, there is a performance implication.

In the Network File System environment, a file is represented as a byte stream. That byte stream is accessible for reading and writing at any byte offset for any byte length. In the MVS environment, a file is represented as a collection of records. The record, rather than a single byte, is the smallest object that can be processed. Therefore, the OS/390 NFS server has to convert the byte stream operations from Network File System clients into standard access method operations on MVS.

To convert byte stream operations to MVS access method operations, the server has to determine which record in the MVS file contains the offset specified in the Network File System read or write request. To determine this, the server reads, mapping byte offsets to records, from the last known location in the file until the record containing the requested byte offset is located.

For example, suppose a file on MVS contains 10,000 variable-length records with a maximum length of 80 bytes for any record. Suppose the first Network File System request received tells the server to read 4,000 bytes starting at offset 10,000 bytes. Because the file has not been opened yet, the server would open the file and start reading at the first record, searching for the record that contains offset 10,000. Once it found the record, the server would process the request, which might involve reading more records to find enough bytes to satisfy the request.

Another complication involved in mapping byte offsets to records is the processing defined by the user to apply to a file. For example, if you specify text mode processing with line terminators, the perceived offset into a file from a given client changes.

Cached Data Writing

The OS/390 NFS might cache writes if out-of-sequence data packets are received or if a block of data is partially filled. If the Network File System is processing in the binary data, the writes will remain cached until one of the following occurs: the timeout for the request has been reached or the number of cached packets exceeds the number specified in **cachwindow()**. If the Network File System is processing text data, the writes remain cached until a timeout occurs. The missing data is padded with binary zeroes and record delimiters so that cached writes for text processing are written in the MVS data set on DASD at the location specified in each cached data packet. In the case of cached data packets for binary processing, only binary zeroes will be used to pad the missing data written at the specified location on DASD. See the Table 2 table below.

Attention: For NFS version 3 COMMIT procedure, OS/390 server will only support committing the cached data when the data set is timed out.

Table 2. Breakdown of Text and Binary Writes

	Binary	**Text**
Data is flushed to DASD when:	The number of cached packets exceeds the amount specified in **cachwindow()**, or the file times out.	The file times out. If the number of packet exceeds the amount specified in **cachwindow()**, all new out of sequence packets will be dropped.
Padding	Binary zeros	Binary zeros
Record delimiters	There are no record delimiters. Therefore, there is no attempt to add end of line characters.	There can be record delimiters. Therefore, an end of line character is added to the end of the record.

Case Sensitivity — maplower, nomaplower

If the processing attribute **maplower** is specified, the MVS file name is mapped to the lower case when returned to the client. If the processing attribute **nomaplower** is specified in the attributes, all entries in the exports data set are case-sensitive. Therefore the client MOUNT request must specify the MVS qualifier with the correct case to successfully match the exports data set entry.

Selecting Text or Binary Processing Modes — text, binary

You can specify either *text* or *binary* processing mode when you access files. This processing mode does not describe the type of data in the original file, but rather, it specifies whether to convert between ASCII and EBCDIC when sending file contents between the MVS host and the client workstation. Refer to "MOUNT Command Syntax and Examples" on page 71 for additional information on text and binary processing of files using the OS/390 NFS client.

Binary Processing Mode

The binary processing mode specifies to send and receive file contents between the MVS host and the client in binary form, avoiding the ASCII/EBCDIC conversion required in text mode. This is faster than text mode. However, users on MVS cannot read the file, because the contents are not in EBCDIC. Therefore, use the binary processing mode to create or access a file only if the file is not intended to be shared with users on the MVS host, or the file content is binary.

When fixed-length records are written in binary mode, the server pads the last record of the file with null characters if the last record is less than the fixed record length. These padding bytes are counted in the file size.

Text Processing Mode

With the text processing mode, when data is read, record boundaries are converted to (or from) workstation line terminators such as `'lf'` or `'crlf'`, and data representation is changed between EBCDIC and ASCII.

Selecting How Blanks Are Handled — blankstrip, noblankstrip: When fixed-length records are written in text mode, records are padded with blanks if the record length is larger than a line, and if the **blankstrip** processing attribute is enabled. (When sending data *from* MVS, *blankstrip* strips trailing blanks. When sending data *to* MVS, *blankstrip pads* the records with blanks). In text mode processing, data representation is changed from ASCII to EBCDIC, and line terminators are converted to record boundaries. All data is converted according to the active translation table. Therefore, if the data set contains a mixture of characters and binary data, binary data is converted as well. In text mode, be careful not to mix your text data (characters) with binary data.

If you are writing data to a fixed-length MVS file in text mode with blank stripping enabled, and the data contains blanks at the end of the line, an I/O error occurs. This is because the server is not able to return the blanks to the client when the file is read back.

If you save a fixed-length MVS file in text mode with one or more lines exceeding the maximum record length, an I/O error occurs. For example, suppose an MVS file has fixed-length records of 80 bytes. After you edit the file using the vi editor on your workstation, one of the file's records is 83 bytes long (exceeding the fixed length by 3 bytes). When you save the file back to the server, the MVS file is either partially or totally destroyed, and the "I/O Error" message appears on your screen.

While you are still in the editing session, save the edited file in an alternate local file. After you correct the local file so that no line exceeds 80 bytes, save it back into the MVS file.

If you get an error message when trying to create or access an MVS file, see "Appendix A. Messages to the Client" on page 103 for further explanation of the message.

Using crlf or lf Line Terminators: For a DOS or OS/2 client, use `'crlf'` as the line terminator sequence when accessing data in text processing mode. For an AIX or UNIX client, use `'lf'` as a line terminator when using text processing mode.

Number Representation

The **text** processing mode does not change the number representation format between the host and client. When you choose **text** as the processing mode, the Network File System converts characters between ASCII format and EBCDIC format, and processes line terminator, but no other translation of user data occurs.

When you choose **binary** as the processing mode, Network File System stores your data unchanged. Therefore, regardless of the processing mode you choose, you cannot change numbers from one client workstation's format to another client workstation's format.

Accessing Migrated Files — retrieve, noretrieve; wait, nowait

Sometimes files on MVS are migrated to another storage level, such as a space-saving format on DASD or tape. If your file has been migrated and you try to access it, it might take awhile for it to be recovered back into primary storage. The **retrieve** and **noretrieve** processing attributes control what happens when you try to access a migrated file.

There are three ways that the retrieve or noretrieve option is controlled.

1. Using the Default Retrieve Attribute

   You can use the default retrieve processing attribute by not entering **retrieve** or **noretrieve** in your MOUNT command or file access command.

2. Specifying Retrieve with the MOUNT Command

   You can issue the MOUNT command, specifying **retrieve** or **noretrieve**. The attributes specified in the MOUNT command override the attributes in the default attribute data set.

   In this example, migrated files under the mount point are not retrieved. However, you can access files under the mount point which are not migrated.

   ```
   $ mount mvshost1:"smith,noretrieve" /u/smith/mnt
   ```

   Conversely, the next command causes the migrated files under the mount point to be retrieved when accessing the files.

   ```
   $ mount mvshost1:"smith,retrieve" /u/smith/mnt
   ```

3. Specifying Retrieve with a File Access Command

   You can issue a file access command with the attribute **retrieve** or **noretrieve** specified. The attributes specification in the file access command overrides the attributes in the MOUNT command and the server default attributes.

   This command causes all files under the mount point `/u/smith/mnt` to be retrieved if they are migrated:

```
$ ls -l "/u/smith/mnt,retrieve"
```

This command, however, does not cause migrated files under the mount point `/u/smith/mnt` to be retrieved:

```
$ ls -l "/u/smith/mnt,noretrieve"
```

Accessing Migrated System-Managed Data Sets

DFSMS/MVS Version 1 Release 3 allows data set attribute accessibility for SMS-managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3. Supported data set types are SMS-managed PS, VSAM ESDS, VSAM KSDS, VSAM RRDS, PDS, and PDSE. Migrated PDS/PDSE members are not supported.

The OS/390 NFS server is able to obtain the attributes of a supported SMS-managed migrated data set without recalling the data set. Attributes such as the time stamp and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS-managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a non-Network File System application (for example, by the TSO editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall must be done to determine the correct file size.

When a request is made to remove any of the supported SMS-managed migrated data sets, the data set will be deleted without recall. For PDS and PDSE migrated data sets, the data set will be recalled in order to read its member information.

File Handle Refresh

File handles of mounted objects (directories or file systems) are saved on DASD in a mount handle data set and are automatically re-established when the server restarts. However, file handles for the files within a mounted object are kept in virtual storage (memory) and they are lost if the server restarts. This results in stale file handles, and the clients must request a new file handle by redoing the lookup on the file.

Mapping Between the Workstation and MVS File Systems

In MVS, a file is called a *data set*, and the two terms are used interchangeably in this book. The OS/390 NFS server presents information to you in the form of a UNIX** (or AIX), OS/2, or DOS file, though the information is actually stored on MVS-owned DASD in the form of an MVS data set.

The files for a computer system are organized into a *file system*. The UNIX, OS/2, and DOS environments use a file system which is a hierarchy of *directories*. MVS, however, uses a non-hierarchical file system in which groups of data sets are referred to by specifying a *high-level qualifier*.

The MVS high-level qualifier can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the high-level qualifier for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the high-level qualifier of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

File Name Extension Mapping

File extension mapping allows users to access members of PDS/PDSE data sets on the MVS host which are mapped from client machine files that contain file extensions. Each PDS/PDSE data set on the host can only be mapped with one unique file extension. For example: `IBMUSER.TEXT(M1)`, `IBMUSER.TEXT(M2)`will map to `m1.txt`, `m2.txt` under directory `ibmuser.text` on the client machine. This capability allows client machine tools such as editors and compilers, to process host files remotely without modification. There are site and processing attributes that are associated with the file extension mapping. The client user can specify the **fileextmap** attribute to turn the file extension mapping on and the **nofileextmap** attribute to turn the file extension mapping off. The **sidefile(***dsname***)** attribute allows the client user to specify a side file name that is different than the default side file name during the MOUNT operation. For example:

```
[C:\] mount z: mvshost1:"user1.pds,sidefile(hlq.nfs.mapping)"
```

The side file specified at the MOUNT command will be searched first followed by the default side file. The system administrator can specify the maximum space available for side files using the new site attribute **sfmax (***n***)**. A sample mapping side file is provided as GFSAPMAP in the NFSSAMP library. Refer to the *OS/390 Network File System Customization and Operation* for more information.

Mounting MVS Data Sets onto a Client Mount Point

To access an MVS file system from the client, you use the MOUNT command to create a temporary link between specific MVS data sets and a UNIX directory (preferably empty) or an unused logical drive. The UNIX directory or drive is called a *mount point*.

You use an MVS high-level qualifier in the MOUNT command to specify which MVS data sets to mount onto a mount point. The MVS data sets beginning with the specified high-level qualifier appear as files under the mount point. See Figure 2 on page 16.

You can also perform a MOUNT using a fully qualified data set name or an alias to a user catalog, but not the catalog name itself. Only Integrated Catalog Facility (ICF) cataloged data sets are supported by the OS/390 NFS server, and tape data sets are not supported.

Data set organizations supported include:
- Physical sequential (PS)
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS
- HFS
- SAM extended format data sets

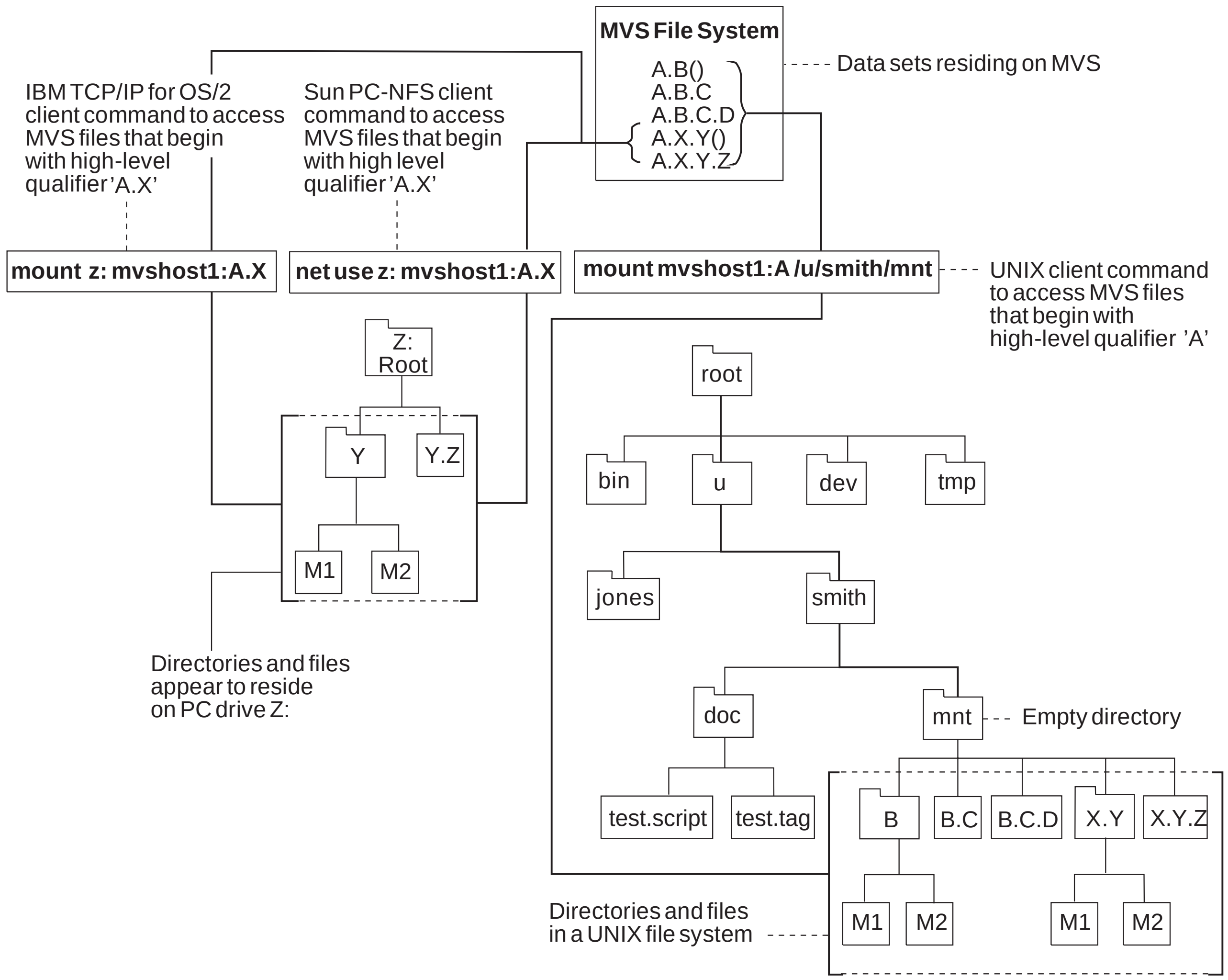
Both SMS-managed and non-SMS-managed data sets are supported.
- Multi-volume data sets are not supported except for extended format data sets.
- Generation data sets are also not supported.

Notes:

1. The filesize for the MVS conventional data set as a directory has a dummy size with a value of 8192.
2. For NFS Version 3 CREATE procedure, OS/390 server does not harden the exclusive create verify token to disk.

Variants of the MOUNT Command

The name of the command that performs the MOUNT operation varies for different client implementations. For example, the SUN** PC-NFS** implementation uses the **net use** command while most UNIX implementations use the MOUNT command.



MVS File System
A.B()
A.B.C
A.B.C.D
A.X.Y()
A.X.Y.Z
Data sets residing on MVS
IBM TCP/IP for OS/2 client command to access MVS files that begin with high-level qualifier 'A.X'
Sun PC-NFS client command to access MVS files that begin with high level qualifier 'A.X'
mount z: mvshost1:A.X
net use z: mvshost1:A.X
mount mvshost1:A /u/smith/mnt
UNIX client command to access MVS files that begin with high-level qualifier 'A'
Z: Root
Y
Y.Z
M1
M2
Directories and files appear to reside on PC drive Z:
root
bin
u
dev
tmp
jones
smith
doc
mnt
Empty directory
test.script
test.tag
B
B.C
B.C.D
X.Y
X.Y.Z
M1
M2
M1
M2
Directories and files in a UNIX file system

Figure 2. Examples of Mounting MVS Data Sets on OS/2, DOS, and UNIX Clients. The OS/2 and DOS clients are mounting the MVS data sets which begin with the high-level qualifier "`A.X`". The UNIX client is mounting the MVS data sets which begin with the more inclusive high-level qualifier "`A`". The parentheses indicate a PDSE containing the members M1 and M2. The PDSEs `A.B()` and `A.X.Y()` appear as directories, and their members appear as files within those directories.

Using a PDS or PDSE as a Directory

If the data sets specified include partitioned data sets, a second level of hierarchy is shown. This allows you to define one level of directories under the mount point. Thus, you can issue **mkdir** to create a directory (stored as a PDS or PDSE) and then create files (stored as members of a PDS or PDSE) within that directory.

This use of a PDSE is shown in Figure 2 on page 16, which illustrates the mapping of file names between client file systems and the MVS file system resulting from a MOUNT command.

Using Multiple Mount Points

You can arrange groups of data sets into several UNIX mount directories (or PC mount drives) by using MVS naming conventions to mount specific data sets at each mount point. For example, you could mount `user1.project1` to get all data sets beginning with “user1.project1” mounted at one point in the local file system, and you could mount `user1.project2` at another point. This would create the effect of two distinct directories (or drives), one containing the `user1.project1.*` data sets, and the other containing the `user1.project2.*` data sets.

Data Set Serialization and Sharing

The OS/390 NFS server handles data set serialization and sharing differently depending on the type of data set:

Physical sequential
The server insures the read/write integrity of a physical sequential data set by SVC 99 dynamic allocation with exclusive option whenever a physical sequential data set is opened for output; otherwise it is allocated with share option.

VSAM data set
The server dynamically allocates a VSAM data set with share option and allows the VSAM access method to manage data sharing via the *shareoptions* specified during data set definition.

PDSE data set
The server dynamically allocates a PDSE data set with share options and allows the PDSE functions to manage the integrity of the PDSE data set and its members.

PDS data set
The server dynamically allocates a PDS data set with share option and surrounds the PDS and its members with exclusive ENQs against the QNAME=SPFEDIT and RNAME=data set name. This does not protect the PDS from other MVS users who are attempting to access the PDS without performing ENQ against SPFEDIT similar to the OS/390 NFS server.

NFS Protocol

Table 3 illustrates that the Network File System procedures are not all supported for conventional MVS data sets.

Table 3. Network File System Procedures

Procedure	Version 2 Protocol	Version 3 Protocol
link	no	no
mknod	N/A	no
readlink	no	no
readdirplus	N/A	no
setattr	yes	yes
statfs	yes	N/A

Table 3. Network File System Procedures (continued)

Procedure	Version 2 Protocol	Version 3 Protocol
symlink	no	no
Note: Setattr only supports filesize=0 truncation and UNIX permission is set to 777.		

File System Size

The server generates values for the UNIX attributes of file system block size and file system size in blocks, because they have no meaning in the MVS context, except for HFS files. Table 4, Table 5, and Table 6 on page 19 illustrate the MVS values that the server generates.

Table 4. File System Values to get Dynamic File System Information (FSSTAT):

Description	Conventional MVS Value
tbytes Total size, in bytes, of the file system	10000000000
fbytes The amount of free space, in bytes, in the file system	8000000000
abytes The amount of space, in bytes, available to the user identified by the authentication in the RPC	80000000
tfiles The total number of file slots in the file system	200000
ffiles The number of free file slots in the file system	20000
afiles The number of free file slots that is available to the user corresponding to the authentication information in the RPC	2000
invarsec The number in seconds for which the file system is not expected to change	0

Table 5. File System Values to get Static File System Information (FSINFO):

Description	Conventional MVS Value
rtmax The maximum number in bytes for the read request supported by the server	65536 (64KB)
rtpref The preferred size of the **read** request	32768 (32KB)
rtmult The suggested multiple for the size **read** request	4096
wtmax The maximum size of a **write** request supported by the server	65536 (64KB)
wtpref The preferred size of the **write** request	32768 (32KB)
wtmult The suggested multiple for the size of a **write** request	4096
dtpref The preferred size of the **readdir** request.	8192
maxfilesize The maximum size of a file on the system	2 ** 63 - 1
time_delta File time using **setattr**	(0,1000000)
Properties	
FSF_LINK	1
FSF_SYMLINK	0
FSF_HOMOGENEOUS	1
FSF_CANSETTIME	1

Table 6. File System Values to Retrieve POSIX Information (PATHCONF):

Description	**Conventional MVS Value**
linkmax Maximum number of hard links	1
name_max Maximum length of a component file name (file name + attributes)	255
no_trunc The server will reject any name that is longer than the name_max	True
chown_restricted To change either the owner or the group associated with the data set	True
case_insensitive The server does not distinguish the case when interpreting file names	True
case_preserving If True, the server file system will preserve the case of a name during a **create**, **mkdir**, **mknod**, **symlink**, **rename**, or **link**	False

Chapter 3. Creating Conventional MVS Data sets

This chapter tells how to create the various types of data sets (files) supported by the OS/390 NFS server.

The examples shown are for the AIX RS/6000 platform. Any examples for other platforms are so indicated. For PC platforms, also refer to "Chapter 7. Commands and Examples for OS/2 Clients" on page 51 and "Chapter 8. Commands and Examples for DOS and Sun PC-NFS Clients" on page 59.

Overriding Data Set Creation Attributes

When you create an MVS file, default file creation attributes are applied, unless you override them. The attributes are passed to the MVS host.

Data set creation attributes are controlled in the following ways, in increasing order of priority:

- Default server data set creation attributes
- Default installation data set creation attributes, specified by the system administrator in the attributes data set
- DFSMS data class attributes
- Data set creation attributes specified in the **mount** (or **net use**) command
- Data set creation attributes specified in the **mkdir**, **vi** (edit), or **cp** (copy) commands (highest priority)

Note: These data class attributes are not supported by the OS/390 NFS server:
- Retention period/Expiration date
- Number of volumes
- VSAM imbed index option
- VSAM replicate index option
- CI size of data component
- Percentage of CI or CA free space

Preparing to Create an MVS File

When creating an MVS file, you should know whether to process the file in text or binary mode (see "Selecting Text or Binary Processing Modes — text, binary" on page 12) and what type of file to create.

These types of files are supported by the OS/390 NFS server:
- Physical sequential (PS)
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS
- SAM extended format data sets

Keyed access to files and GDG are not supported.

Naming MVS Files

The OS/390 Network File System uses comma *(,)* as a delimiter to a list of file attributes. Do not use *(,)* as a special character in file name. For example:

```
$ vi "/u/smith/new,text"
```

This indicates to OS/390 Network File System that a file called *new* is being edited in the attribute **text** mode, not file *new,text*.

When naming conventional MVS files, you must follow the MVS file naming conventions, as described in *DFSMS/MVS Using Data Sets*.

For information about the OS/390 UNIX System Services naming conventions refer to "Chapter 4. Using OS/390 UNIX System Services Files" on page 29.

An MVS file name (or data set name) can consist of one or several simple names joined so that each represents a level of qualification. For example, the MVS file name `DEPT58.SMITH.DATA3` is composed of three qualifiers.

The following characteristics apply to the MVS file name:

- Each qualifier consists of 1 to 8 alphanumeric characters, national characters (@, #, $), or a hyphen (-).
- Each qualifier must start with an alphabetical or national character.
- The period (.) separates simple names from each other.
- Including all simple names and periods, the length of the MVS file name must not exceed 44 characters.
- PDS and PDSE member names can be up to 8 characters long.

For information about the MVS file system, see "Mapping Between the Workstation and MVS File Systems" on page 14.

Restrictions Using Alias Names for MVS Files

For non-VSAM files, alias names can be used interchangeably with the true file name except on remove (**rm** and **rmdir**) and rename (**mv**) requests. Renaming or removing an alias name results in an I/O error. This is due to an MVS restriction.

If the true name of an MVS file is renamed or removed and alias names have been defined for the file, MVS deletes the alias names during execution of the rename or remove request.

Creating Physical Sequential Files

When creating a physical sequential (PS) file, specify the **dsorg(ps)** attribute (if it is not the default already) with the **mount** command or a file creation command, such as the **vi** UNIX (or AIX) command.

1. Create a local directory on your client to be used as a mount point. For example (with UNIX):

   ```
   $ mkdir /u/smith/mnt
   ```

2. Mount the MVS file system. For example, suppose your host is `mvshost1`, and you want to issue a mount on the high-level qualifier `smith`. You could enter:

```
# mount mvshost1:"smith,dsorg(ps)"  /u/smith/mnt
```

If you do not specify any other attributes, the MVS site defaults are used. You can use the **showattr** command to display the site defaults.

3. You can use the **vi** UNIX command to create the `new` file:

```
$ vi /u/smith/mnt/new
```

When you save the file using **vi**, you have just created a new MVS PS file named `SMITH.NEW`.

You could get the same results by specifying **dsorg(ps)** in the file creation command rather than in the **mount** command:

```
# mount mvshost1:smith /u/smith/mnt
$ vi "/u/smith/mnt/new,dsorg(ps)"
```

Creating Direct Access Files

When creating a direct access (DA) file, specify the **dsorg(da)** attribute (if it is not the default already) with the **mount** command or a file creation command (such as the **vi** UNIX command).

1. Create a local directory on your client to be used as a mount point. For example (with UNIX):

```
$ mkdir /u/smith/mnt
```

2. Mount the MVS file system. For example, suppose your host is `mvshost1`, and you want to issue a mount on the high-level qualifier `smith`. You could enter:

```
# mount mvshost1:"smith,dsorg(da)"  /u/smith/mnt
```

If you do not specify any other attributes, the MVS site defaults are used. You can use the **showattr** command to display the site defaults.

3. Next, you can use the **vi** UNIX command to actually create the `new` file:

```
$ vi /u/smith/mnt/new
```

You have just created a new MVS DA file named `SMITH.NEW`.

You could get the same results by specifying dsorg(da) in the file creation command rather than in the **mount** command:

```
# mount mvshost1:smith /u/smith/mnt
$ vi "/u/smith/mnt/new,dsorg(da)"
```

Creating PDSs and PDSEs

Partitioned data sets (PDSs) and partitioned data sets extended (PDSEs) can be used as directories, and their members are files within those directories. An illustration of the use of PDSs to act as directories is shown in Figure 2 on page 16. For general information on PDSs and PDSEs, see*DFSMS/MVS Using Data Sets* .

You cannot create new directories within a PDS or PDSE, due to the nature of these data structures.

Creating a PDS or PDSE — mkdir dsntype(pds), dsntype(library)

To create a PDS or PDSE, perform the following steps:

1. Create a local directory on your client to be used as a mount point. For example (with UNIX):

   ```
   $ mkdir /u/smith/mnt
   ```

2. Mount the MVS file system (accessing files that begin with the high-level qualifier of `smith`):

   ```
   # mount mvshost1:"smith,mgmtclas(normal)"
   /u/smith/mnt
   ```

3. If creating a PDSE, use the **mkdir** (make directory) UNIX command specifying the **dsntype(library)** attribute to create a PDSE named `smith.datalib`:

   ```
   $ mkdir /u/smith/mnt/"datalib,dsntype(library)"
   ```

   If creating a PDS, use the **mkdir** (make directory) UNIX command specifying the **dsntype(pds)** attribute as follows:

   ```
   $ mkdir /u/smith/mnt/"datalib,dsntype(pds),dir(20)"
   ```

   Omitting dsntype(pds): You can omit specifying the **dsntype(pds)** attribute if **pds** has been specified for the **dsntype** attribute either in your site attribute data set or in your mount point.

4. You can use the **vi** UNIX command to create a PDS or PDSE member named `smith.datalib(member1)`:

   ```
   $ vi "/u/smith/mnt/datalib/member1,text"
   ```

   Input your text, save it, and quit.

You have now created a PDS or PDSE member, which is processed in text processing mode. You can use the **cat** UNIX command to view the contents of your PDS or PDSE member.

Note: OS/390 NFS server supports a maximum of 14,562 members in a PDS or PDSE data set. When a Network File System read-directory request on a PDS or PDSE is processed, the OS/390 NFS server will return up to 14,562 member names. Other requests, such as read and write, to individual members are not affected.

Removing a PDS or PDSE — rm, rmdir

To remove a PDS or PDSE, first make sure that the PDS or PDSE is empty. You can delete all members under the directory using the **rm** UNIX command. Then use the **rmdir** (remove directory) UNIX command. This example removes the `datalib` directory, and confirms its removal by a failed try to query it (**ls** is the UNIX list files command):

```
$ ls -F /u/smith/mnt/datalib
data1* data2* data3*
$ rm /u/smith/mnt/datalib/*
$ rmdir /u/smith/mnt/datalib
$ ls -F /u/smith/mnt/datalib
/u/smith/mnt/datalib not found
```

Accessing PDS or PDSE Members

There is more than one way to mount and access PDS and PDSE members. For example, you could display the existing PDS member `smith.source(bigblue)` by entering either of these command sequences:

```
$ mkdir /mnt
# mount hostname:"smith.source,text" /mnt
$ cat /mnt/bigblue
```

or

```
$ mkdir /mnt
# mount hostname:"smith,text" /mnt
$ cat /mnt/source/bigblue
```

These two approaches are equivalent.

Updating or extending a PDS or PDSE Member

The OS/390 NFS server does not support updating or extending a PDS or PDSE member directly. To update or extend a PDS or PDSE member, a client program must follow these steps:

1. Copy the file to the client machine
2. Update or extend the copied version on the local system
3. Truncate the original MVS file to zero size by sending a SETATTR request with zero file size
4. Copy the updated version on the local host to MVS by writing request

Some client editors follow the above steps, for example, OS/2 EPM and KEDIT editors, and the AIX and UNIX vi editor. Other editors do not follow the above steps, for example, OS/2 E editor and the OS/390 UNIX OEDIT editor. In the latter case the user must save the updated version into a new file.

Timing Out While Writing a PDS or PDSE Member

If you are writing to a PDS or PDSE member and a timeout occurs, the timeout causes the member to close. The remaining write requests appear to append to a PDS or PDSE member. The write request does not complete successfully and causes an I/O error. To avoid timing out, increase the time on the timeout setting.

Wildcard Copy to a PDS or PDSE

To ensure that a wildcard copy of a PDS or PDSE is completed successfully, the PDS or PDSE member must be closed and dequeued (if necessary). For example, using the statement below, a wildcard copy will fail if any member inside of `smith.datalib` is open.

```
$ cp smith.*  /u/smith/mnt/datalib
```

Limitations of a PDS

The PDS support in the Network File System adheres to the conventions used in MVS. For example, you cannot have more than one member of a PDS open for output at a time. If you try to CREATE **|** REMOVE **|** RENAME **|** WRITE a member of a PDS while another member is open for output, you get a “Permission denied” message.

A PDS member stays open for the timeout period specified in the appropriate timeout processing attribute, or until you try to create or write to another member.

Concurrent Writes to a PDSE

The Network File System supports concurrent writes to a PDSE. If you are writing to one member of a PDSE, another client can write to any other member in the same PDSE. You can also use ISPF to edit a PDSE member while another Network File System client is writing to a different member of the same PDSE.

Creating VSAM Files

The Network File System supports three types of VSAM files: key-sequenced (KSDS), entry-sequenced (ESDS), and relative record (RRDS). However, keyed access and relative-number access to the files are not supported.

If you plan to update a VSAM data set (for example, with the **vi** editor or with the **cp** copy command), the data set must have been defined with the REUSE option. Trying to write back a VSAM data set that was not defined as reusable results in an “I/O error”, “failure to open”, or similar error message. If you create a VSAM file using the Network File System, the REUSE option is used by the server.

For more information on VSAM files, see *DFSMS/MVS Using Data Sets*.

In the following example, the attributes indicate that:
- Spanned records are allowed
- Organization is key-sequenced
- Keys are 8 bytes long and start in position 0 of each record
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT

```
$ cp ksds.old "ksds.new2,spanned,dsorg(indexed),keys(8,0),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example for creating a VSAM ESDS file, the attributes indicate that:
- Spanned records are allowed
- Organization is entry-sequenced
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT

```
$ cp esds.old "esds.new3,spanned,dsorg(nonindexed),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example, the attributes indicate that:
- Spanned records are not allowed
- Organization is relative record, numbered in ascending order
- Average record size is 1024
- Maximum record size is 1024
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT

```
$ cp rrds.old "rrds.new4,nonspanned,dsorg(numbered),
   recordsize(1K,1K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

Exploiting SAM Striped Files

With SAM striping, data I/O is done in parallel to improve performance. For a file with 16 stripes, data is simultaneously processed on the first track of the allocated space in each of the 16 volumes. This allows for quick access to all the information.

The OS/390 NFS server can support data set striping through the use of data class and storage class attributes that define extended format data sets. The OS/390 NFS server can exploit the performance of extended format data sets by reading multiple blocks at a time when reading ahead.

For more information on striped files, see *DFSMS/MVS Using Data Sets*.

Chapter 4. Using OS/390 UNIX System Services Files

This chapter explains what you need to know to access OS/390 UNIX files from a client workstation. This chapter explains:

- The hierarchical file system (HFS)
- POSIX compatibility
- Attributes specific to OS/390 UNIX
- Protecting your OS/390 UNIX files
- Accessing OS/390 UNIX files from the client
- Linking an MVS data set to a hierarchical file system
- UNIX look and feel

Note: For detailed information about OS/390 UNIX refer to *OS/390 UNIX System Services User's Guide*, order number SC28-1891.

HFS File System

OS/390 UNIX provides a hierarchical file system (HFS) for MVS. A file within OS/390 UNIX is called an HFS file. HFS files are organized in a hierarchy of files and directories on a tree much like UNIX. A directory can contain files or other sub-directories. The highest level directory is called the *root directory*. Figure 3 on page 30 shows an example of mounting an HFS directory from a UNIX client.

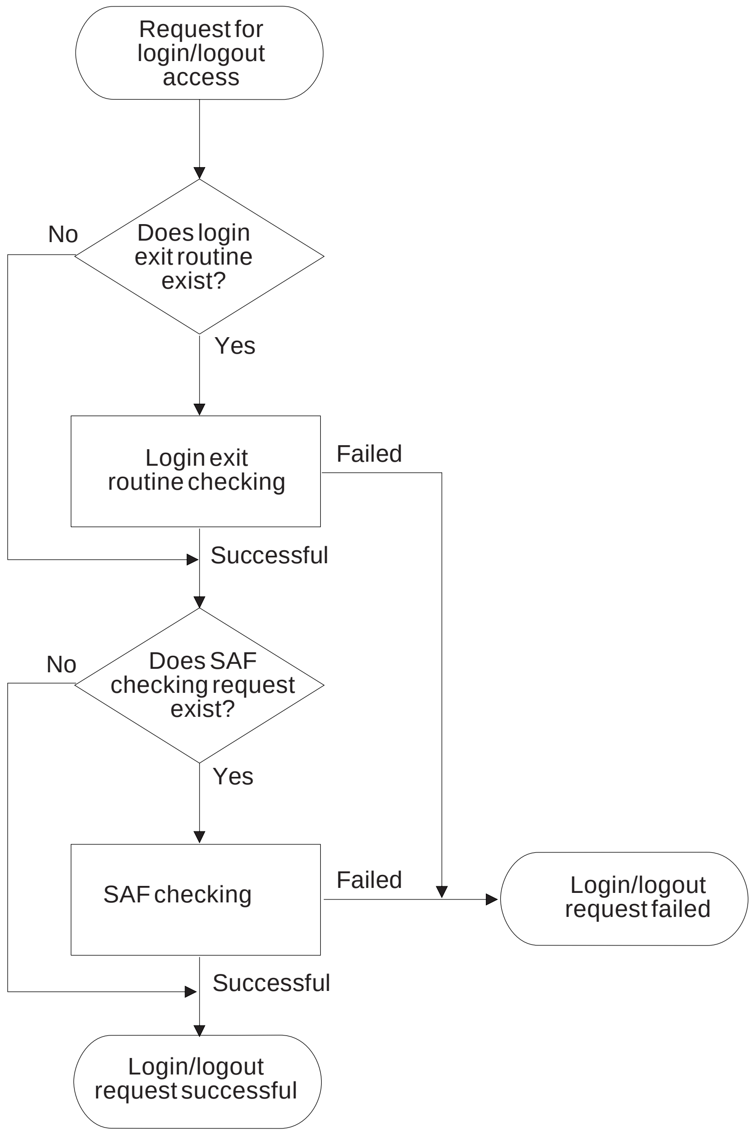

Request for login/logout access
Does login exit routine exist?
No
Yes
Login exit routine checking
Failed
Successful
Does SAF checking request exist?
No
Yes
SAF checking
Failed
Login/logout request failed
Successful
Login/logout request successful

Figure 3. Example of Mounting an HFS File from a UNIX Client

An HFS file system must be mounted by an MVS system operator using a TSO **MOUNT** command before that HFS file system can be mounted by an Network File System client via the OS/390 NFS server.

HFS files are byte-oriented rather than record-oriented (unlike conventional MVS data sets). This data can be shared with TSO OS/390 UNIX users in addition to Network File System clients. All data written to an HFS file can be read by all programs as soon as it is written. You can also copy data between HFS and MVS data sets using OS/390 UNIX utilities like ISHELL.

Text or Binary Processing

HFS is a byte-oriented, hierarchical, EBCDIC file system. The OS/390 NFS server provides ASCII to EBCDIC text translation. If you are just using the mainframe as a repository for your workstation (ASCII) data, you should use the binary mode to speed processing. If you use text mode, data from your workstation is converted into EBCDIC when it is stored on the mainframe. Conversely, when the OS/390 NFS server returns the data to your client system, it converts the data back into

ASCII. The conversion can slow processing, but might be necessary if you are sharing data with other MVS users. All data is converted according to the active translation table. Therefore, if the data set contains a mixture of characters and binary data, binary data is converted as well. In text mode, then, be careful not to mix your text data (characters) with binary data.

In text mode, you can either use the OEMVS311 translation table or a customized translation table to convert data between ASCII and EBCDIC. If you are using OS/390 UNIX and text mode processing, specify the OEMVS311 translation table with the xlat processing attribute. The OEMVS311 table translates ASCII (ISO 8859-1) to and from EBCDIC (1047 - OS/390 UNIX System Services). TCP/IP for MVS version 3.1 provides the OEMVS311 table. This table translates the UNIX line terminator (lf) to the OS/390 UNIX line terminator (nl). Refer to *TCP/IP for MVS: Customization and Administration Guide*, “Using Translation Tables”, to see how to create and customize your own translation tables.

This is an example of specifying the OEMVS311 translation table during a mount:

```
$  mount 1stc3mvs:"/hfs/usr/man/C,text,xlat(oemvs311)" /mnt
$  export MANPATH=/mnt
$  man more
```

In this example:

1stc3mvs
Name of the OS/390 UNIX host

/hfs HFS prefix

/urs/man/C
HFS directory to be mounted

text Specifies that data be converted between ASCII and EBCDIC

xlat(oemvs311)
Specifies that the translation table named OEMVS311 is used to convert data between ASCII and EBCDIC

/mnt Local mount point

man more
Obtains a Man Page description of the More command.

POSIX Compatibility

The Network File System supports file access to the OS/390 UNIX file system. OS/390 UNIX supports a set of standards called the Portable Operating System Interface (POSIX). See *OS/390 UNIX System Services User’s Guide* for information on POSIX compliance. With OS/390 UNIX, the Network File System:
- Supports hierarchical directories
- Allows file names up to 255 characters in length
- Allows path names up to 1023 characters in length
- Supports mixed-case names and special characters, except nulls, slashes, and commas in file and path names
- Supports UNIX-style file access permissions
- Supports group ID and user ID at the file level
- Supports the full NFS protocol (including external links)
- Enables data sharing between clients and the OS/390 UNIX
- Enables you to link conventional MVS data sets to a POSIX path name

This support incorporates the basic strengths of the MVS/ESA system for both existing MVS data and applications and for new POSIX conforming data and applications.

NFS Protocol

The OS/390 UNIX are compliant with all of the OS/390 Network File Systems version 2 and version 3 protocols.

Attributes Specific to OS/390 UNIX System Services

The following attributes are specific to the OS/390 UNIX:

- **sync** and **async** processing attributes for version 2 protocol only
- **hfs** site attribute
- **extlink** attribute, refer to "Linking an MVS Data Set to a Hierarchical File System" on page 34.

Note: These attributes are also explained in "Chapter 10. Descriptions of Attributes for the OS/390 NFS Server" on page 89.

Synchronous Write to an HFS File for NFS version 2 Protocol

Use the **sync** and **async** processing attributes to specify whether data received by a write request for an HFS file object is committed to nonvolatile storage before the write response is returned to you.

If **sync** is specified for an HFS file object, the data is written to HFS.

For greater throughput, you can alternatively specify **async**. Your data is then committed to the disk some time after the write request is received from the Network File System client. Your data is written to disk when the write timeout occurs, or if OS/390 UNIX reclaims buffer cache storage.

The **sync** and **async** processing attributes only apply to HFS data access. They are ignored for MVS data set access. A TSO OS/390 UNIX user doesn't have to wait for the data to be committed to nonvolatile storage before accessing. OS/390 UNIX maintains a central buffer cache and a TSO OS/390 UNIX user can access the data as if it were in the file once it has been written by the OS/390 NFS server.

Synchronous Write to an HFS File for NFS version 3 Protocol

For the *write* procedure there is a processing argument **stable** and output parameter **commit** which specifies whether data received by a write request for an HFS file object is committed to nonvolatile storage before the write response is returned to you.

If the **stable** processing argument is used during the write procedure, there are three modes when the write procedure writes to a file:

- **file_sync** The OS/390 NFS server must commit all data written and all file system data to stable storage before returning commit results.
- **data_sync** The OS/390 NFS server must commit all data written and sufficient metadata to enable retrieval of data, before it returns a reply to the client.
- **unstable** The OS/390 NFS server may not commit any part of the data and metadata to stable storage, before returning a reply to the client. The data will be committed when a timeout occurs.

For the **commit** procedure, the OS/390 server will support committing the entire data and metadata to stable storage.

Authorization Checking When Writing to an HFS File

There is no support of OPEN and CLOSE of a file since SUN NFS Protocol is a stateless model. In order to support the OPEN and CLOSE semantics, the OS/390 NFS server allows the *owner* of an HFS file to write permission regardless of the UNIX permission bits setting on the file.

HFS Site Attribute

To access OS/390 UNIX files, you must know the HFS prefix defined by your system administrator (the default is /hfs). You can use the **showattr** command to display the HFS prefix defined for your location. You use this prefix in your mount command before the path name of the OS/390 UNIX directories that you are mounting. The HFS prefix is used by the NFS server to distinguish OS/390 UNIX directories from conventional MVS data sets, but the HFS prefix isn't part of the path name that you see. After you have entered the mount command, you access HFS files using the local mount point.

Protecting Your OS/390 UNIX System Services Files

As an OS/390 UNIX user, you can control the read, write, and execute access to your files by other users in and outside of your group by setting the permission bits associated with the files.

To access OS/390 UNIX files from the Network File System, you must be defined as an OS/390 UNIX user. The system programmer defines you as an OS/390 UNIX user by assigning an *OS/390 UNIX user ID (UID)* and an *OS/390 UNIX group ID (GID)* to you. The UID and GID are numeric values associated with a TSO/E user ID. The values are set in the RACF user profile and group profile when you are authorized to use OS/390 UNIX. The system uses the UID and GID to identify the files that you can access. Your specific UID value identifies you as a user of OS/390 UNIX services. A GID value is a unique number assigned to a group of related users. These numbers appear in the RACF user profile. See *OS/390 UNIX System Services Planning* for more information.

Accessing OS/390 UNIX System Services Files from a Client

Most of the commands used to access OS/390 UNIX files are identical to the commands used to access conventional MVS data sets. These commands are:
- mvslogin
- showattr
- mount
- umount
- mvslogout

The only command that is changed for OS/390 UNIX is the **mount** command.

Note: The syntax of the commands listed above may vary between platforms, refer to the appropriate chapter for examples specific to the platform you are using to access OS/390 UNIX files.

If you are using AIX (or any other UNIX-based operating system) refer to "Chapter 6. Commands and Examples for AIX and UNIX Clients" on page 39.

If you are using OS/2, refer to “Chapter 7. Commands and Examples for OS/2 Clients” on page 51.

If you are using DOS, refer to “Chapter 8. Commands and Examples for DOS and Sun PC-NFS Clients” on page 59.

mount Examples

Table 7 shows how to mount OS/390 UNIX files from various platforms.

mvshost1	Name of the MVS host
/hfs	HFS prefix
/smith	HFS directory to be mounted
/u/smith/mnt	Local mount point

Table 7. Examples of the mount Command for Clients

Clients	Command Examples
AIX, UNIX, Solaris	mount mvshost1:″/hfs/smith″ /u/smith/mnt
OS/2	mount z: mvshost1:/hfs/smith
DOS	mount z: mvshost1:/hfs/smith
Sun PC-NFS	net use z: mvshost1:/hfs/smith

Note: The /hfs prefix value is used by the OS/390 NFS server to determine if a file is an OS/390 UNIX file, and does not appear in the path name of an HFS file once it is mounted.

Linking an MVS Data Set to a Hierarchical File System

This section explains how to access an MVS data set via an HFS path name by using the external link command. It also explains how to display the contents of an external link and how to delete an external link.

Creating an External Link

You can create an external link to an MVS data set, and then transparently access the MVS data set by referencing the external link. The external link simulates a UNIX-like hierarchical naming convention for conventional MVS data sets. This is done using the **ln** command. For example:

```
mount mvshost1:USER1 /mnt
mount mvshost1:/hfs/u/nfs /samples
ln -s USER1.MVSFILE /samples/linkfile,"extlink"
```

In this example an HFS file object, /linkfile, of the file type ″extlink″ is created containing the file name of the MVS data set **USER1.MVSFILE** to be accessed. The source file must be mounted to an HFS file system. The external link must reference an MVS data set. All future references to **/samples/linkfile** access **USER1.MVSFILE** transparently.

In this example the file **/usr/pub/myfile** is copied to the MVS data set **USER1.MVSFILE** contained in the external link **/samples/linkfile**:

```
cp /usr/pub/myfile /samples/linkfile
```

Your installation should make sure that the appropriate security permissions have been obtained to access the MVS data set. You will receive ″Permission Denied″ message if the mount point `/mnt` has not been established on USER1.

A mount point must be established before the external link is established. Otherwise, the error code *ACCESS DENIED* is returned. For physical sequential data sets, the high level qualifier of a data set must be established. For example, if you had a file called `smith.test.data` you can mount with `smith`, `smith.test`, or `smith.test.data` as your high level qualifier. For PDS and PDSE data sets, the fully qualified name must be established as a mount point. An example of a fully qualified name would be, `smith.test.data`.

Displaying the Contents on an External Link

You can display the contents of an external link by appending the ″extlink″ sequence to the external link path name. This permits the user to inspect the contents of the external link with the **ls -l** command.

This example shows how to display the attributes and contents of the external link **/samples/linkfile**:

```
ls -l /samples/linkfile,"extlink"
lrwxrwxrwx  1 user1 13 Jun 17 20:43 /samples/linkfile ->USER1.MVSFILE
```

This example shows how to display the attributes of the MVS target data set `USER1.MVSFILE`:

```
ls -l /samples/linkfile
-rw-rw-rw-  1 root  2112 Sep 28 13:50 /samples/linkfile
```

Deleting an External Link

The external link file object is deleted with the **remove** request:

```
rm /samples/linkfile,"extlink"
rm /samples/linkfile
```

Either rm command results in the HFS external link file object alone being removed. The target MVS data set is not affected.

UNIX Look and Feel

Using the OS/390 NFS with OS/390 UNIX managed files provides UNIX client users with a transparent view of their data. The file attributes are maintained in the same way as is found on any UNIX system:
- Regular, directory, link, device, and FIFO file types
- User, group, and other read/write/execute access permissions
- UID and GID file ownership
- File size

To access HFS files, it is necessary to be defined to RACF as an OS/390 UNIX user. Some installations might prefer to provide users with unrestricted access to their OS/390 UNIX data by specifying `security(none)` or `security(exports)` in the site attributes. With this setting, the client's user ID and group ID credentials are used for all file access authentication, and there is no requirement for the user to be defined to RACF or to perform the mvslogin command.

Note: For `security(none)` and `security(exports)` options, the UID of the ROOT (UID=0 from the workstation) is mapped to UID of NOBODY (UID=65534) by the OS/390 NFS server. The implication is that the OS/390 NFS server will use the mapped UID of 65534 for all HFS file authorization checking. For example, file creation owner UID is set to 65534 in the HFS file attribute.

File System Size

For HFS files, see Table 8 and Table 9 for the following file system values that are returned:

Table 8. File System Values to get Static File System Information (FSINFO):

Description	HFS Value
rtmax The maximum number in bytes for the read request supported by the server	65536 (64KB)
rtpref The preferred size of the **read** request	32768 (32KB)
rtmult The suggested multiple for the size **read** request	4096
wtmax The maximum size of a **write** request supported by the server.	65536 (64KB)
wtpref The preferred size of the **write** request	32768 (32K)
wtmult The suggested multiple for the size of a **write** request	4096
dtpref The preferred size of the **readdir** request.	8192

Table 9. File System Values to Retrieve POSIX Information (PATHCONF):

Description	HFS Value
linkmax Maximum number of hard links	2 ** 31
name_max Maximum length of a component file name (file name + attributes)	255
no_trunc The server will reject any name that is longer than the name_max	True
chown_restricted To change either the owner or the group associated with the data set	False
case_insensitive The server does not distinguish the case when interpreting file names	False
case_preserving If True, the server file system will preserve the case of a name during a **create**, **mkdir**, **mknod**, **symlink**, **rename**, or **link**	True

Chapter 5. The OS/390 NFS NLM and the OS/390 NSM

This chapter provides an overview of the OS/390 Network File System Network Lock Manager (OS/390 NFS NLM) and the OS/390 Network File System Network Status Monitor (OS/390 NFS NSM). It explains how they work together to provide file locking and access control capability over OS/390 NFS. In addition, this chapter also explains:

- Monitored Lock
- Non-Monitored Locks
- Locking Files
- Locking Records

The OS/390 NFS NLM

The OS/390 NFS NLM allows a client on the host to lock a record or a file on the OS/390 NFS server. A client user can either choose to lock the entire file or a record section of a file. The two types of locks that the client host uses are monitored locks and non-monitored locks.

The OS/390 NFS NLM only supports *advisory locking*. Advisory locking is when the operating system keeps track of which files have been locked by which process, but does not prevent a process from writing to a file that is locked by another process. This means that a process can ignore an advisory lock if the process has adequate permission.

Monitored Locks

Monitored locks provide the client user with reliability. If the server host on which the monitored locks are established fails, the locks are reinstated when the server host recovers. The locks that are held by the client host are discarded by the OS/390 NFS NLM on the server host if the client host fails before the locks are released. Monitored locks will only work correctly if both the server host and the client host are running NSM.

Non–Monitored Locks

Non-Monitored locks are used on personal computer (PC) operating systems. Non–Monitored locks provide the same functionality as the monitored locks with one exception. If the server host on which the locks are established, fails and recovers, the locks will not be re–established. The client host is responsible for detecting a server host failure and re–establishing the locks. In addition, the client host informs the OS/390 NFS NLM when it has rebooted so that the client host can discard all of the locks and file shares held for the client.

Locking Records

A record is a byte-stream range. The record starts at the current offset and extends forward for a positive size value and backwards for a negative size value. If the size value is zero, the record affected extends forward from the current offset through the end of the file.

In order to lock a record, the client user can use the *lockf* function. The *lockf* function has four values. The F_ULOCK value unlocks a previously locked region. The F_LOCK value locks a region. The F_TLOCK value tests and locks a region. The F_TEST value tests a region to see if it is locked. When the client user uses the F_LOCK function when the region is being accessed by another function, the F_LOCK function waits for the region to become available and then perform the locking function. This is called *blocking*. When the client user uses the F_TLOCK function and the region that it is being locked is being used by another function, *lockf* returns with a value of –1 and *errno* set to EAGAIN or EACCESS. This is called *nonblocking*. By using the F_TEST function, the client user can check to see if a lock is set, without setting a lock.

If the file starts with the size value of zero and the *lockf* system call is used, then the entire file is locked. In order to issue a byte range lock, the client user must specify a small program that specifies the section of the file that they are locking. In "Appendix E. Sample Program for LOCKD and STATD" on page 121, the client user is issuing a byte range lock.

Locking a file

The *flock* system call is used to lock and unlock a file. The *flock* system call uses four constants to lock a file. The LOCK_SH constant specifies shared locks while the LOCK_EX specifies exclusive locks. The LOCK_UN constant unlocks the file and the LOCK_NB specifies that the file should not be blocked when it is locked. The LOCK_NB can be used with both the shared locks and the exclusive locks.

More than one shared lock can be applied to a file but the file cannot have both a shared lock and an exclusive lock at the same time. Furthermore, a file also cannot have multiple exclusive locks at one time. If the lock is successful, zero is returned by *flock*. If the lock is unsuccessful, then a —1 is returned. When the lock fails because the file is already locked and a *nonblocking* lock was requested, the *errno* is set to EWOULDBLOCK. See "Appendix E. Sample Program for LOCKD and STATD" on page 121.

The OS/390 NFS NSM

The OS/390 NFS NSM is a service that provides applications with information on the status of network host. Each OS/390 NFS NSM keeps track of its own ″state″ and notifies any interested parties of a change in its ″state″.

For correct operation of the OS/390 NFS NSM, the client and the server hosts are required to monitor each other. When a lock request is issued by a process running on the client host, the NLM on the client host request the NSM on the client host to the monitor the server host. The client NLM then transmits the lock request to the OS/390 NFS NLM on the server. On reception of the lock request the OS/390 NFS NLM on the server host will request the OS/390 NFS NSM on the server host to monitor the client host. In this way, each host is monitored by the NSM on the other host.

Chapter 6. Commands and Examples for AIX and UNIX Clients

This chapter gives the syntax and examples of commands that AIX RS/6000 users need to know to access MVS data sets from a client. Some examples are also provided for HP/UX, UNIX, Sun Solaris and SunOS environments. This chapter shows how to:
- Log on to MVS from your client
- Access MVS files from your client
- Display default mount point attributes
- Query mount points
- Unmount MVS files from the client
- Log out of MVS

The **mount** and **umount** commands are operating system specific commands. They are not shipped with OS/390 NFS. See your Network File System client documentation for the exact syntax and usage.

Using Commands on AIX

Here is a summary of the syntax for the commands described in this chapter:

```
►►──mvslogin──┬─────────────────┬──┬───────────┬──┬─────────────┬──────────►
              ├─-p──────────────┤  └─-g──group─┘  └─-a──account─┘
              ├─-n──────────────┤
              ├─-pn─────────────┤
              └─-P──mvs_passwd──┘

►─hostname──┬────────────┬──────────────────────────────────────────────►◄
            └─mvs_userid─┘
```

-p Causes a prompt for your MVS password. The password is passed to MVS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n Causes a prompt for a new password.

-pn
Causes a prompt for the user's current password and then causes two prompts for the user's new password.

-P *mvs_passwd*
No prompt for your MVS password; just type your MVS password after the -P. This enables you to automate your MVS login.

-g *group*
A group name string passed to MVS for accounting purposes. The maximum length is 8 characters.

-a *account*
An account string passed to MVS for accounting purposes. The maximum length is 16 characters.

hostname
The name of the MVS host (for example, `mvshost1`)

mvs_userid
A user ID that MVS recognizes as valid. If you do not specify this parameter,

your workstation user name is used. The OS/390 NFS server does not support the use of an alias user ID or a mixed case user ID with the **mvslogin** command.

```
►►──mount──┬──────────────┬──hostname:"mvs_qual──────────────────────────►
           └─-o clnt_opt──┘

    ┌──────────────────┐
    ▼                  │
►───┬──────────────┬───┴──"──/localpath─────────────────────────────────►◄
    └─,attribute───┘
```

-o *clnt_opt*
The client **mount** command options (such as `soft,timeo=20`). Refer to the documentation of your client operating system for a description of the options for your client environment.

hostname
The name of the MVS host (for example, `mvshost1`)

mvs_qual
The path name of an OS/390 UNIX file (must begin with the /HFS prefix) or the high-level qualifier(s) of a conventional MVS data set that you are accessing in the MVS file system (such as `smith` or `smith.project`).

attribute
A OS/390 NFS server data set creation or file processing attribute (such as `text`). See “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89. If you specify any attributes, make sure you enclose *mvs_qual* and the attributes in double quotation marks.

/localpath
The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

```
►►──showattr──┬─────┬──hostname──┬──────────────┬─────────────────────────►◄
              └─-t──┘            └─/localpath───┘
```

-t Used to specify tersed output.

hostname
The name of the MVS host (for example, `mvshost1`)

/localpath
The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

```
►►──umount──/localpath─────────────────────────────────────────────────────►◄
```

/localpath
The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

```
►►──mvslogout──hostname──────────────────────────────────────────────►◄
```

hostname
The name of the MVS host (for example, `mvshost1`)

Quick Reference of AIX and UNIX Commands

This screen is an example of a standard MVS login and logout procedure for AIX RS/6000. In this example:

smith MVS user ID and high level qualifier for MVS data sets

mvshost1
System name of MVS host

/u/smith/mnt
The name of the mount point

GFSA*nnnt*
Messages starting with GFSA apply towards OS/390 NFS requests. These messages are explained in “Appendix A. Messages to the Client” on page 103.

```
# mvslogin mvshost1 smith
Password required
GFSA973I Enter MVS password: password
GFSA955I smith logged in ok.
# mount mvshost1:"smith" /u/smith/mnt
mount: mvshost1:"smith"
"smith" was attached successfully.
# umount /u/smith/mnt
Unmounting '/u/smith/mnt' ...    successful
# mvslogout mvshost1
UID 200 logged out ok.
```

You can use the mount command with no parameters specified to list the mount points on your client system.

Accessing OS/390 UNIX System Services and Conventional MVS

To access OS/390 UNIX files or conventional MVS data sets, enter both the **mvslogin** command to log into the MVS host system and the **mount** command to mount the files or data sets to your local system. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to **saf** or **safexp**. Once the files or data sets are mounted to a local directory, you can read, write, create, delete, and treat the mounted files as part of your workstation’s local file system. When you are finished with your work, use the **umount** and **mvslogout** commands to break the connection. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

To access files on MVS systems where the OS/390 NFS server site *security* attribute is set to **saf**, **exports**, or **safexp**, you need an MVS user ID and password, and authorization to access the files that you need. You can only establish one MVS session for each MVS user ID. If you do not already have an MVS user ID, an MVS password, and access authorization, request them from your MVS system administrator.

Note: If you cannot use the **mvslogin**, **mvslogout**, or **showattr** commands, they might be installed incorrectly or in another directory. Ensure that your system administrator has made the executable code for these three commands available to your workstation and that you have been given the correct path name to find the commands. Also, make sure that your version of these commands matches the release of the OS/390 NFS that you are using. Otherwise, the commands might not function properly.

mvslogin Command Examples

Use the **mvslogin** command to log in to MVS from your workstation. The **mvslogin** command can be issued multiple times and the last one overrides the previous one. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to saf or safexp.

Table 10 shows examples of the **mvslogin** command where `mvshost1` is the name of the MVS host and `smith` is the user's ID on MVS.

Table 10. Examples of the mvslogin Command for Clients

Command Examples
`mvslogin -p mvshost1 smith`
`mvslogin -P smithspw -g finance -a 5278 mvshost1 smith`
`mvslogin -n mvshost1 smith`
`mvslogin -pn -a 5278 mvshost1 smith`
`mvslogin mvshost1`
`mvslogin mvshost1 smith`

In the example where the user enters **mvslogin mvshost1**, the current login client user ID is used as the MVS user ID.

In the example where the user enters **mvslogin mvshost1 smith**, the system then prompts for Smith's MVS password. If Smith logs in successfully, this message appears:

```
GFSA955I smith logged in ok.
```

Otherwise, an appropriate error message appears.

Note: Messages with the prefix of GFSA and GFSC apply to Network File System requests. These messages are further explained in “Appendix A. Messages to the Client” on page 103.

When an OS/390 UNIX UID or GID segment is defined with the user identification, an additional check is done to compare the OS/390 UNIX UID or GID with the client UID or GID during the login processing. An informational message is returned when the server and the client UID or GID do not match. This informational message contains the OS/390 UNIX UID and GID for the MVS user identification.

Note: The authentication is considered successful even if the UID and GID do not match. The message is issued for the user's information only.

For the PCNFSD authentication request, the server UID and GID is returned to the client user if the UID and GID are defined. Otherwise, an arbitrary number is generated and returned to the client user.

“Permission Denied” Message

If you have successfully logged in and get the “Permission denied” message while trying to access data, that can be due to one of the following cases:

- An **mvslogout** command for the same MVS host has been entered from the same client platform. See **mvslogout** for details.
- Your MVS user ID has been automatically logged out because the **logout** attribute value has been exceeded. This can happen when you leave the client idle for too long. Enter the **mvslogin** command again, and start your processes again. To find out how many seconds you can stay logged in while your client is idle, issue the **showattr** command and look at the **logout** attribute.
- Another mvslogin to the same MVS host using the same MVS ID has been entered from the same UID and the same client platform. If this is the case, retry your access.

Note: Some clients give a somewhat different message such as “Access is Denied”.

mount Command Examples

Use the **mount** command to make a connection between a mount point on your local file system and one or more files in the MVS file system.

```
# mount mvshost1:"smith" /u/smith/mnt
```

mvshost1
The name of the host server

smith The name of the high-level qualifier of the MVS files

/u/smith/mnt
The name of the mount point (preferably an empty directory)

At the same time, you can also predefine some of the attributes for the mount point, overriding the default attributes.

Overriding the Default Attributes

To override the default attributes, specify different attributes with the **mount** command or in a file access or creation command (such as **vi** in AIX or UNIX).

There are two kinds of attributes that you can modify:

Data set creation attributes provide information about an MVS file to the OS/390 NFS server, such as:

- The type of file
- How the file is allocated

Note: Data set creation attributes do not apply to OS/390 UNIX files.

Processing attributes provide information to the OS/390 NFS server about how to handle the file. For example:

- How long the files remain open
- Whether the file contents are sent and received in text form, or in binary form to avoid ASCII/EBCDIC conversion

Use the **showattr** command to display the default data set creation and processing attributes. For descriptions of the attributes, see “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89.

Files are created and processed using the mount point attributes that were in effect when the files were last mounted. If your installation’s default attributes have been changed (via the **exportfs** operand of the **MODIFY** system operator command or restart of the server) and you want to apply these new default attributes, you can unmount and mount again (using the **umount** and **mount** commands).

When you access the file with a data access or creation command, you can override some of the attributes that were set by a **mount** command or the server default settings.

Using mount to Override Server Default Attributes

In this example, the **mount** command is used to modify two processing attributes, specifying **binary** (rather than **text**) and **readtimeout(30)** (rather than the server default **readtimeout** value):

```
# mount mvshost1:"smith,binary,readtimeout(30)" /u/smith/mnt
```

mvshost1
The name of the host server.

smith The name of the high-level qualifier of the MVS files.

binary The processing mode for file contents sent between MVS and the client (binary mode avoids ASCII/EBCDIC conversion).

readtimeout(30)
The amount of time (30 seconds) allowed since the last read access before the file is closed.

/u/smith/mnt
The name of the mount point (preferably an empty directory)

Getting Authorization to Access Files

If the mount fails, check with your system administrator to ensure that you are authorized to access the MVS data sets or OS/390 UNIX files and that the data sets or files are listed in the exports data set. The privilege level required to enter **mount** and **umount** commands varies among client operating system implementations. Many UNIX implementations limit these commands to the root user or superuser mode. If the MVS system operator issues the **freeze=on** operand of the **MODIFY** command, all new tries to mount an MVS or HFS data set fail until the MVS system operator issues the **freeze=off** operand. If the OS/390 UNIX system operator issues the **freeze=onhfs** operand of the **MODIFY** command, conventional MVS data sets can still be mounted, but all new tries to mount OS/390 UNIX files fail until the system operator issues the **freeze=offhfs** operand.

Saving of Mount Points

Once the **mount** command is issued successfully and a mount point is established between a local directory and the MVS or OS/390 UNIX file system, the mount point information is saved in the mount handle data set by the OS/390 NFS server. This information is used to automatically reestablish active mount points when the server is started. When accessing conventional MVS data sets, a mount point is active if the last activity against this mount point is less than the **restimeout** attribute value set by the system administrator (restimeout does not apply for HFS file system access). The **mount** command does not need to be reissued for the same mount point in further sessions unless the **umount** command has been used to disconnect the mount point or the mount point has been disconnected by the cleanup activity of the **restimeout** site attribute. For more information about the **restimeout** site attribute refer to “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89.

Automatic Timed Logout — logout Attribute

If there is no activity on the client within the period specified in the **logout** attribute of the attributes file, or the server stops, the connection between the server and the client workstation is logged out automatically. You must issue the **mvslogin** command again to get access to the MVS files.

Displaying Default and Mount Point Attributes — showattr

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point. If you specify a mount point, **showattr** shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89.

If you omit the *hostname*, you must specify the */localpath*.

Table 11 shows examples of the **showattr** command for some clients.

Table 11. Examples of the showattr Command for Clients

Client Environments	Command Examples
AIX, UNIX	**`showattr mvshost1 /u/smith/mnt`**
OS/390	**`showattr mvshost1 /u/smith/mnt`**
SunOS	**`showattr mvshost1 /u/smith/mnt`**
Solaris	**`showattr mvshost1 /u/smith/mnt`**

Make sure that your version of the showattr command matches the release of the OS/390 NFS that you are using. Otherwise, the OS/390 NFS server attributes do not display.

These examples show different ways you can use the **showattr** and **mount** commands.

Figure 4 shows a **showattr** command with just the host name (`mvshost1` in this example) specified. The default attributes for the server are displayed.

```
$ showattr mvshost1

OS/390 NFS Server Data Set Creation Attributes:

lrecl(80)               recfm(fb)               blksize(3200)
space(10,10)            blks                    dsorg(ps)
dir(5)                  unit()                  volume(nfs003)
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)
mgmtclas()              dsntype(pds)            norlse
dataclas()              storclas()

OS/390 NFS Server Processing Attributes:

text                    lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30)
sync                    nofileextmap            xlat(OEMVS)
sidefile(f066440.nfs.mapping)

OS/390 NFS Server Site Attributes (not modifiable):

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,8)           restimeout(48,0)        hfs(/hfs)
bufhigh(32M)            readaheadmax(16K)       cachewindow(128)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(none)               nopcnfsd                security(exports,exports,exports)
leadswitch               sfmax(2)               nochecklist
public(IBMUSER.WEBNFS,/HFS/)
```

Figure 4. Displaying Default Attributes

If you use the terse (-t) option, the attributes display like this:

```
$ showattr -t mvshost1
lrecl(80),recfm(vb),blksize(3200),space(10,10),blks,dsorg(ps),dir(5),unit(),
vol(nfs003),recordsize(512,4K),keys(64,0),nonspanned,shareoptions(1,3),
mgmtclas(),dsntype(pds),norlse,dataclas(),storclas()
text,lf,blankstrip,nofastfilesize,retrieve,maplower,mapleaddot,executebitoff,
setownerroot,attrtimeout(120),readtimeout(90),writetimeout(30),sync,nofileextmap
xlat(OEMVS),sidefile(f066440.nfs.mapping)
mintimeout(1),nomaxtimeout,logout(1800),nfstasks(8,8),restimeout(48,0),
hfs(/hfs),bufhigh(32M),readaheadmax(16K),cachewindow(128),percentsteal(20),
maxrdforszleft(32),logicalcache(16M),smf(none),nopcnfsd,
security(exports,exports,exports),leadswitch,sfmax(2),nochecklist,
public(IBMUSER.WEBNFS,/HFS/)
```

Figure 5 on page 47 illustrates the **showattr** command being used to display the attributes for the MVS host named `mvshost1` as well as the mount point, `/u/smith/mnt`.

Figure 5 on page 47 illustrates the specified options. In the second **showattr** command, the client user specifies `/u/smith/mnt` in addition to `mvshost1`. This shows the users specified settings at that mount point, rather than the default settings in the attributes data set.

```
# mount mvshost1:"smith,text,space(5,0),trks" /u/smith/mnt
$ showattr mvshost1

OS/390 NFS Server Data Set Creation Attributes:

lrecl(8196)            recfm(vb)              blksize(0)
space(100,10)          blks                   dsorg(ps)
dir(27)                unit()                 volume()
recordsize(512,4K)     keys(64,0)             nonspanned
shareoptions(1,3)
mgmtclas()             dsntype(pds)           norlse
dataclas()             storclas()

OS/390 NFS Server Processing Attributes:

binary                 lf                     blankstrip
nofastfilesize         retrieve               maplower
mapleaddot             executebitoff          setownerroot
attrtimeout(120)       readtimeout(90)        writetimeout(30)
sync                   nofileextmap           xlat(OEMVS)
sidefile(hlq.nfs.mapping)

OS/390 NFS Server Site Attributes (not modifiable):

mintimeout(1)          nomaxtimeout           logout(1800)
nfstasks(8,8)          restimeout(48,0)       hfs(/hfs)
bufhigh(32M)           readaheadmax(16K)     cachewindow(128)
percentsteal(20)       maxrdforszleft(32)     logicalcache(16M)
smf(userfile)          nopcnfsd     security(exports,exports,exports)
leadswitch             sfmax(0)               checklist
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Figure 5. Displaying Default and Mount Point Attributes (Part 1 of 2). The client user changed the **space(100,10)**, **blks**, and **binary** attributes to **space(5,0)**, **trks**, and **text** for the mount point `/u/smith/mnt`, and then specified that mount point in the second **showattr** command.

```
$ showattr mvshost1 /u/smith/mnt

OS/390 NFS Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(5,0)             trks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)
mgmtclas()              dsntype(pds)            norlse
dataclas()              storclas()

OS/390 NFS Server Processing Attributes:

text                    lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30)
sync                    nofileextmap            xlat(OEMVS)
sidefile(hlq.nfs.mapping)

OS/390 NFS Server Site Attributes (not modifiable):

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,8)           restimeout(48,0)        hfs(/hfs)
bufhigh(32M)            readaheadmax(16K)       cachewindow(128)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(userfile)           nopcnfsd     security(exports,exports,exports)
leadswitch              sfmax(0)                checklist
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Figure 5. Displaying Default and Mount Point Attributes (Part 2 of 2). The client user changed the **space(100,10)**, **blks**, and **binary** attributes to **space(5,0)**, **trks**, and **text** for the mount point `/u/smith/mnt`, and then specified that mount point in the second **showattr** command.

Unmounting and Logging Out of MVS

This section describes the **umount** and **mvslogout** (or **mvslogut**) commands.

Disconnecting Your Mount Point — umount

Use the **umount** command to break the connection between the mount point on your client and the server. When you issue this client command, the file you were editing is released (written to DASD). You do not need to unmount after each session, unmount only when you no longer have a need to access the MVS file system. Check the documentation for your client operating system to ensure that you enter the **umount** command correctly.

Table 12 shows examples of the **umount** command for some clients:

u/smith/mnt
The mount point on the local file system

mvshost1
The name of the MVS host system

Table 12. Examples of the umount Command for Clients

Client Environments	umount Command Examples
AIX, UNIX	**umount /u/smith/mnt**
SunOS	**umount mvshost1**

Table 12. Examples of the umount Command for Clients (continued)

Client Environments	umount Command Examples
Solaris	**umount mvshost1**

For example, suppose that you want to unmount from the server, and the mount point on your workstation is named `/u/smith/mnt`. You could enter the **umount** command as follows:

```
# umount /u/smith/mnt
```

"No Such File or Directory" Message The MVS system operator can also unmount your workstation from the server. If this happens before you try to unmount, you get a "No such file or directory" error message.

Ending Your MVS Session — mvslogout, mvslogut

Use the **mvslogout** or **mvslogut** command to disconnect from the MVS host. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

An **mvslogout** to an MVS user ID cancels a prior **mvslogin** to the same MVS user ID from the same local host.

Your account is automatically logged out if it is inactive for the period of time specified in the **logout** site attribute.

Table 13 shows an example of the **mvslogout** command for some clients, in which the name of the MVS host is `mvshost1`.

Table 13. Example of the mvslogout Command for Clients

Client Environments	mvslogout Command Examples
AIX, UNIX	**mvslogout mvshost1**
HP/UX	**mvslogout mvshost1**
SunOS	**mvslogout mvshost1**
Solaris	**mvslogout mvshost1**

If you log out successfully, a message like this appears:

```
GFSA958I uid 215 logged out ok.
```

Chapter 7. Commands and Examples for OS/2 Clients

This chapter tells you how to enter commands to access files residing on MVS from an OS/2 client. It also provides examples of the commands. This chapter tells you how to:
- Access MVS data sets from OS/2
- Log on to MVS from OS/2
- Query mount points
- Display default mount point attributes
- Override default mount point attributes
- Log off and end your MVS session from OS/2

The **mount** and **umount** commands are operating system specific commands. They are not shipped with the OS/390 OS/390 NFS. See the OS/2 Network File System client documentation for the exact syntax and usage.

Using Commands on OS/2

These are the commands you use with IBM TCP/IP for OS/2:

```
►►──mvslogin──┬─────────────────┬──┬────────────┬──┬──────────────┬────────►
              ├─-p──────────────┤  └─-g──group──┘  └─-a──account──┘
              ├─-n──────────────┤
              ├─-pn─────────────┤
              └─-P──mvs_passwd──┘

►──hostname──┬────────────┬───────────────────────────────────────────────►◄
             └─mvs_userid─┘
```

-p Causes a prompt for your MVS password. The password is passed to MVS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n Causes a prompt for a new password. Your security procedures determine whether you need to use this parameter.

-pn
Causes a prompt for your current password and then causes two prompts for your new password. Your security procedures determine whether you should use this parameter.

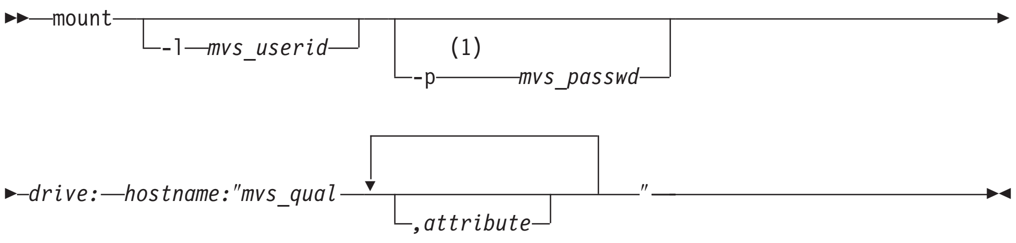
-P *mvs_passwd*
No prompt for your MVS password; just type your MVS password after the -P. This enables you to automate your MVS login. Your security procedures determine whether you should use this parameter.

-g *group*
A group name string passed to MVS for accounting purposes. The maximum length is 8 characters.

-a *account*
An account string passed to MVS for accounting purposes. The maximum length is 16 characters.

hostname
The name of your MVS host (for example, `mvshost1`)

mvs_userid
A user ID that MVS recognizes as valid. If you do not specify this parameter, your workstation user name is used. The OS/390 OS/390 NFS server does not support the use of an alias user ID or a mixed case user ID with the **mvslogin** command.



Notes:

1 No space between this keyword and variable (for example, -lsmith).

-l*mvs_userid*
For PCNFSD, used to specify an MVS user ID.

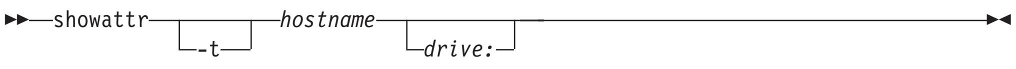
-p*mvs_passwd*
For PCNFSD, used to specify an MVS password.

drive
Specifies the drive to which Network File System attaches the mounted file system.

hostname
The name of your MVS host (for example, `mvshost1`)

mvs_qual
The path name of an OS/390 UNIX file or directory (beginning with the /HFS prefix) or the name of the high-level qualifier of the MVS files that you are accessing in the MVS file system (such as `mvsuser.project` or `smith`).

attribute
A OS/390 Network File System server data set creation or processing attribute (such as `text`). See“Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89. If you specify any attributes, make sure you enclose *mvs_qual* and the attributes in double quotation marks.



-t Use to specify tersed output.

hostname
The name of your MVS host (for example, `mvshost1`)

drive
Specifies the drive to which Network File System attaches the mounted file system.

```
►►──umount──drive:──────────────────────────────────────────────────►◄
```

drive
Specifies the drive to which Network File System attaches the mounted file system.

```
►►──mvslogut──hostname──────────────────────────────────────────────►◄
```

hostname
The name of your MVS host (for example, `mvshost1`)

CONFIG.SYS File Settings

If PCNFSD is not running, you should have statements in your CONFIG.SYS file specifying the client name (called HOSTNAME, meaning “local host”), user ID (UID), and group ID (GID). For example:

```
SET  HOSTNAME=MOHLER
SET  UNIX.UID=-2
SET  UNIX.GID=1
```

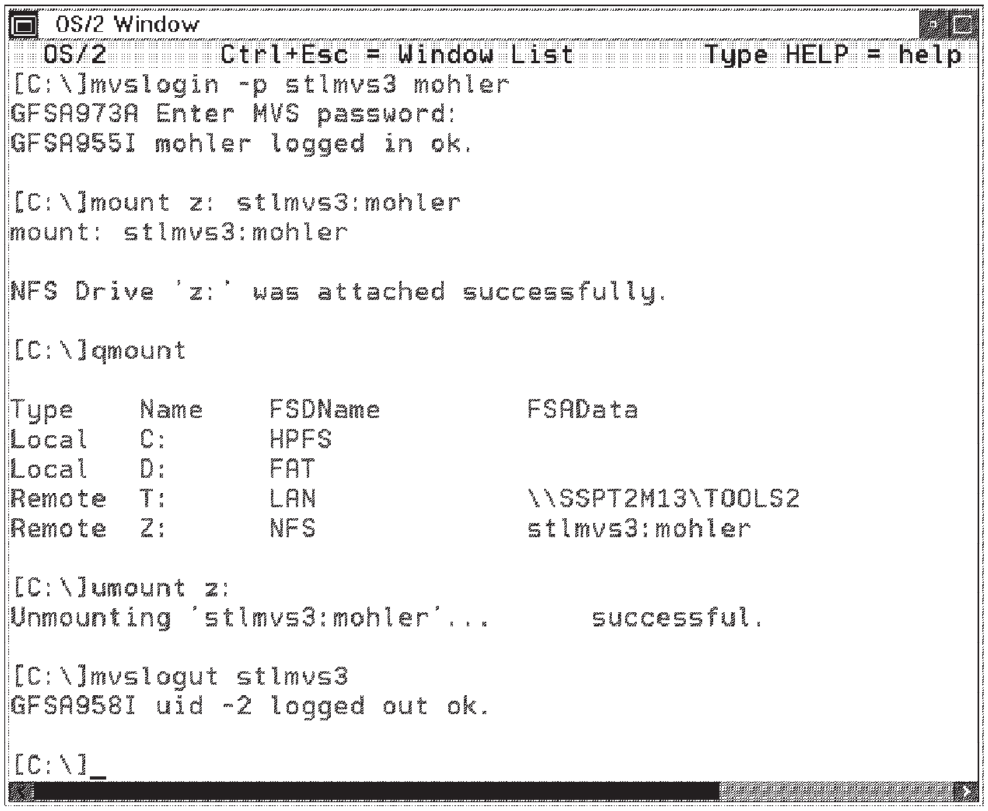
These are UNIX-style credentials, which can be passed from your workstation to the server.

Quick Reference of OS/2 Examples

Figure 6 on page 54 is a quick example of a standard MVS login, mount and MVS logout procedure. In this example:

mohler
MVS user ID and high level qualifier for MVS data sets

stlmvs3
System name of MVS host



```
  OS/2          Ctrl+Esc = Window List          Type HELP = help
[C:\]mvslogin -p stlmvs3 mohler
GFSA973A Enter MVS password:
GFSA955I mohler logged in ok.

[C:\]mount z: stlmvs3:mohler
mount: stlmvs3:mohler

NFS Drive 'z:' was attached successfully.

[C:\]qmount

Type    Name    FSDName         FSAData
Local   C:      HPFS
Local   D:      FAT
Remote  T:      LAN             \\SSPT2M13\TOOLS2
Remote  Z:      NFS             stlmvs3:mohler

[C:\]umount z:
Unmounting 'stlmvs3:mohler'...      successful.

[C:\]mvslogut stlmvs3
GFSA958I uid -2 logged out ok.

[C:\]_
```

Figure 6. Command Examples for OS/2

Accessing OS/390 UNIX System Services and Conventional MVS

To access OS/390 UNIX files or conventional MVS data sets, enter the **mvslogin** command to log onto the MVS host system and the **mount** command to mount the files or data sets to your local system. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to saf or safexp. Once you've mounted the data sets to a local drive, you can read, write, create, delete and otherwise treat the mounted files as part of the workstations local file system. When you are done with your work, use the **umount** and **mvslogut** commands to break the connection. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

If you have PCNFSD client support on your workstation and a PCNFSD request is part of the mount command, you only need to issue the **mount** command, not the **mvslogin** command.

To access files on MVS systems where the OS/390 NFS server site *security* attribute is set to saf, exports, or safexp, you need an MVS user ID and password, and authorization to access the files that you need. You can only establish one MVS session for each MVS user ID. If you do not already have an MVS user ID, an MVS password, and access authorization, request them from your MVS system administrator.

Note: If you cannot use the **mvslogin**, **mvslogut**, or **showattr** commands, they might be installed incorrectly or in another directory. Ensure that your system administrator has made the executable code for these three commands available to your workstation and that you have been given the correct path name to find the commands.

mvslogin Example

Use the **mvslogin** command to log into MVS from the client. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to saf or safexp. Suppose your MVS user ID is `smith`, and the name of the MVS system that you want to log onto is `mvshost1`. You would enter:

```
C> mvslogin mvshost1 smith
```

After you have logged on to the MVS host system, you must issue a **mount** command before you can access MVS files.

mount Example

Use the **mount** command to make a connection between a mount point on your local file system and one or more files in the MVS file system. Suppose the name of the MVS system that you want to log onto is `mvshost1`, and the high-level qualifier of the files that you want to access or create is `smith`. If you are using IBM TCP/IP for OS/2 and you want to use your `Z:` drive to mount the MVS files, you would enter:

```
C> mount z: mvshost1:smith
```

After you have mounted an MVS file or directory, use standard OS/2 commands to access or create MVS files.

Querying Mount Points Example

To view your active mount points you can enter:

```
C> qmount

Type    Name  FSDName  FSAData
Local   C:    FAT
Local   D:    FAT
Remote  Z:    NFS      mvshost1:smith
```

This provides you with a list of active mount points and host names.

showattr Example

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point. Suppose you want to see the values of the OS/390 Network File System server attributes associated with the mount point of the `Z:` drive. To do this, enter:

```
C> showattr mvshost1 z:
```

where `mvshost1` is the name of the MVS host system and `Z:` is the logical drive you used to mount the files. The data set creation attributes, processing attributes, and site attributes are all displayed. For more information about these attributes refer to “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89.

Overriding the Default Attributes

To override the default attributes, specify the attributes you want to change during the mount operation or in a file access or creation command. This section shows examples of how to override default attributes during a mount operation.

Modifying Data Set Creation Attribute Example

In this example the mount command modifies a data set creation attribute, specifying the creation of a direct access (DA) data set (instead of the server default data set organization). You can do this by entering:

```
C> mount z: mvshost1:"smith,dsorg(da)"
```

All subsequent data sets created on this mount point are direct access (DA) data sets. For more information on creating different types of MVS data sets refer to "Chapter 3. Creating Conventional MVS Data sets" on page 21.

Modifying Processing Attributes Example

In this example the mount command modifies two processing attributes, specifying **text** (rather than **binary**) and **readtimeout(30)** (instead of the server default **readtimeout** value). You can do this by entering:

```
C> mount z: mvshost1:"smith,text,readtimeout(30)"
```

The attributes **text** and **readtimeout(30)** override the default values. All files processed during this mount use these processing modes.

umount Example

Use the **umount** command to break the connection between the local mount point and the host system. When you issue this command the file you were editing is released and written to DASD. Suppose you are finished working with a set of MVS data sets that you had mounted to your `Z:` drive, and you are ready to unmount them. You would enter:

```
C> umount z:
```

mvslogut Example

Use the **mvslogut** command when you finish accessing files that reside on a specific MVS system or host. Invoking the **mvslogut** command prevents other users on your workstation from having access to files on that specific MVS system. You should issue the **mvslogut** command at the close of each session. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

For example, if the name of the MVS system that you had logged onto is `mvshost1`, you would log off the MVS system by entering:

```
C> mvslogut mvshost1
```

Note: MVS logout is spelled *mvslogut* because of the 8 character filename convention in OS/2.

Using PCNFSD Login and Authentication

When you issue the **mount** command on an OS/2 platform which is enabled as a PCNFSD client, a PCNFSD authentication request establishes the connection between your client and the server. You should not issue the **mvslogin** or **mvslogut** commands, unless your client doesn't maintain the UID and GID for the mount point base.

mount Example

This is an example of the **mount** command used with PCNFSD.

```
C> mount -l smith z: mvshost1:smith
```

In this example, `smith` is the MVS user ID, `z:` is the drive, `mvshost1` is the name of the MVS system, and `smith` is the high-level qualifier of the MVS files being accessed. The system responds to this **mount** command by asking for the MVS password.

showattr Example

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point. Suppose you want to see the values of the OS/390 Network File System server attributes associated with the mount point of the `Z:` drive. To do this, enter:

```
C> showattr mvshost1 z:
```

where `mvshost1` is the name of the MVS host system and `Z:` is the logical drive you used to mount the files. Figure 7 on page 58 illustrates an example of the attributes that are displayed.

```
OS/390 NFS Server Data Set Creation Attributes:

lrecl(8704)             recfm(fb)               blksize(8704)
space(50,10)            blks                    dsorg(ps)
dir(5)                  unit()                  volume(vm5u18)
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)
mgmtclas()              dsntype(pds)            norlse
dataclas()              storclas()

OS/390 NFS Server Processing Attributes:

text                    lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30)
sync                    nofileextmap            xlat(OEMVS)
sidefile(hlq.nfs.mapping)

OS/390 NFS Server Site Attributes (not modifiable):

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,8)           restimeout(1,14)        hfs(/hfs)
bufhigh(32M)            readaheadmax(32K)       cachewindow(128)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(userfile)           nopcnfsd                security(exports,exports,exports)
leadswitch              sfmax(0)                checklist
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Figure 7. OS/390 NFS Server Data Set Creation Attributes

umount Command and Timing Out

The MVS user ID might be logged off due to a Network File System timeout, when the number of seconds specified in the **logout** attribute is reached, or the server might shut down and start up again. If your MVS user ID, which was authenticated by a PCNFSD request via a **mount** command, is logged off, you need to issue the **umount** and **mount** commands to log on again, because the mount point still exists.

Chapter 8. Commands and Examples for DOS and Sun PC-NFS Clients

This chapter gives the syntax and examples of commands that you need to know to access files residing on MVS from your DOS client. This chapter shows how to:
- Log on to MVS from your client
- Access MVS files from your client
- Display default mount point attributes
- Unmount MVS files from the client
- Log out of MVS

In a DOS environment, you can use:
- IBM TCP/IP for DOS
- Sun PC-NFS

The **mount**, **umount**, and **net use** commands are operating system specific commands. They are not shipped with the OS/390 NFS. See your Network File System client documentation for the exact syntax and usage.

Command Summary for IBM TCP/IP for DOS

You can use these commands with IBM TCP/IP for DOS:

```
►►──mvslogin──┬──────────────────┬──┬───────────┬──┬─────────────┬──────────►
              ├─-p───────────────┤  └─-g──group─┘  └─-a──account─┘
              ├─-n───────────────┤
              ├─-pn──────────────┤
              └─-P──mvs_passwd───┘

►─hostname──mvs_userid──────────────────────────────────────────────────────►◄
```

-p Causes a prompt for the user’s MVS password. The password is passed to MVS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n Causes a prompt for a new password. Your security procedures determine whether you should use this parameter.

-pn
Causes a prompt for the user’s current password and then causes two prompts for the user’s new password. Your security procedures determine whether you should use this parameter.

-P *mvs_passwd*
No prompt for your MVS password; just type your MVS password after the -P. This enables you to automate your MVS login. Your security procedures determine whether you should use this parameter.

-g *group*
A group name string passed to MVS for accounting purposes. The maximum length is 8 characters.

-a *account*
An account string passed to MVS for accounting purposes. The maximum length is 16 characters.

hostname
The name of the MVS host (for example, `mvshost1`)

mvs_userid
A user ID that MVS recognizes as valid. If you do not specify this parameter, your workstation user name is used. The OS/390 NFS server does not support the use of an alias user ID or a mixed case user ID with the **mvslogin** command.

```
►►──mount──┬──────────┬──┬──────────┬───────────────────────────────────────────►
           └─-u──uid──┘  └─-g──gid──┘

►──┬─────────────────────────────────────────┬──drive:──hostname:"mvs_qual──────►
   │                                    (1)  │
   └─-l──mvs_userid──┬────────────────┬──────┘
                     └─-p──mvs_passwd─┘

   ┌──────────────┐
►──▼─┬──────────┬─┴──"──────────────────────────────────────────────────────►◄
     └─,attribute─┘
```

Notes:

1 No space between this keyword and variable (for example, -lsmith).

-u *uid*
Specifies the user ID (UID) on the systems not running PCNFSD.

-g *gid*
Specifies the group ID (GID) on the systems not running PCNFSD.

-l*mvs_userid*
For PCNFSD, used to specify an MVS user ID.

-p*mvs_passwd*
For PCNFSD, used to specify an MVS password.

drive
Specifies the drive letter to which the mounted file system is attached.

hostname
The name of the MVS host (for example, `mvshost1`)

mvs_qual
The path name of an OS/390 UNIX file including the /hfs prefix (such as /hfs/smith/project) or the name of the high-level qualifier of the MVS data sets that you are accessing in the MVS file system (such as `mvsuser1` or `mvsuser1.test`). The file or data set specified must already exist in the MVS catalog.

attribute
A OS/390 Network File System server data set creation or processing attribute (such as `text`). See “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89. If you specify any attributes, make sure you enclose *mvs_qual* and the attributes in double quotation marks.

```
►►──showattr──┬────┬──hostname──┬────────┬──────────────────────────────►◄
              └─-t─┘            └─drive:─┘
```

-t Used to specify tersed output.

hostname
The name of the MVS host (for example, `mvshost1`)

drive
Specifies the drive letter to which the mounted file system is attached.

```
►►──umount──drive:──────────────────────────────────────────────────────►◄
```

drive
Specifies the drive letter to which the mounted file system is attached.

```
►►──mvslogut──hostname──────────────────────────────────────────────────►◄
```

hostname
The name of the MVS host (for example, `mvshost1`)

CONFIG.SYS File Settings

If PCNFSD is not running, you should have statements in your CONFIG.SYS file specifying the client name (called HOSTNAME, meaning “local host”), user ID (UID), and group ID (GID). For example:

```
SET  HOSTNAME=PADILLA
SET  UNIX.UID=200
SET  UNIX.GID=1
```

These are UNIX-style credentials, which can be passed from your workstation to the server.

mvslogin Example

Suppose your MVS user ID is `smith`, and the name of the MVS system that you want to log onto is `mvshost1`. You would enter:

```
C> mvslogin mvshost1 smith
```

After you have logged on, you also need to issue a **mount** command before you can access MVS files.

Note: If you cannot use the **mvslogin**, **mvslogut**, or **showattr** commands, they might be installed incorrectly or in another directory. Ensure that your system administrator has made the executable code for these three commands available to your workstation and that you have been given the correct path name to find the commands.

mount Example

Suppose the name of the MVS system that you want to log onto is `mvshost1`, and the high-level qualifier of the files that you want to access or create is `test`. To use your `Z:` drive to mount the MVS files, you would enter:

```
C> mount z: mvshost1:test
```

After you have mounted an MVS file or directory, use standard DOS commands to access or create MVS files as you would a local file.

Suppose the default for the processing mode is binary, but you want to mount the `test` files using the text processing mode. You can do this by entering:

```
C> mount z: mvshost1:"test,text"
```

showattr Example

Suppose you want to see the values of the file attributes associated with the OS/390 Network File System server. To do this, you could enter:

```
C> showattr mvshost1 z:
```

where `mvshost1` is the name of the MVS system, and `Z:` is the drive that you used to mount the MVS files.

Figure 8 illustrates an example of the data set creation attributes, processing attributes, and site attributes displayed.

```
OS/390 NFS Server Data Set Creation Attributes:

lrecl(8704)             recfm(fb)               blksize(8704)
space(50,10)            blks                    dsorg(ps)
dir(5)                  unit()                  volume(vm5u18)
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)
mgmtclas()              dsntype(pds)            norlse
dataclas()              storclas()

OS/390 NFS Server Processing Attributes:

text                    lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30)
sync                    nofileextmap            xlat(OEMVS)
sidefile(hlq.nfs.mapping)

OS/390 NFS Server Site Attributes (not modifiable):

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,8)           restimeout(1,14)        hfs(/hfs)
bufhigh(32M)            readaheadmax(32K)       cachewindow(128)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(userfile)           nopcnfsd                security(exports,exports,exports)
leadswitch              sfmax(0)                checklist
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Figure 8. OS/390 NFS Server Data Set Creation Attributes

You can modify the data set creation attributes and the processing attributes using the **mount** command, but only the MVS system administrator can change the values of the site attributes.

umount Example

Suppose you are finished working with a set of MVS files that you had mounted to your `Z:` drive, and you are ready to unmount them. You would enter:

```
C> umount z:
```

mvslogut Example

Use the **mvslogut** command when you finish accessing files that reside on a specific MVS system or host. Invoking the **mvslogut** command prevents other users on your workstation from having access to files on that specific MVS system. You should issue the **mvslogut** command at the close of each session.

For example, if the name of the MVS system that you had logged onto is `mvshost1`, you would log off the MVS system by entering:

```
C> mvslogut mvshost1
```

Using PCNFSD Authentication with IBM TCP/IP for DOS

With PCNFSD support, when you issue the **mount** command, a PCNFSD authentication request establishes the connection between your client and the server, so you should not issue the **mvslogin** or **mvslogut** commands.

mount Example

This is an example of the **mount** command used with PCNFSD:

```
C> mount -lsmith z: mvshost1:test
```

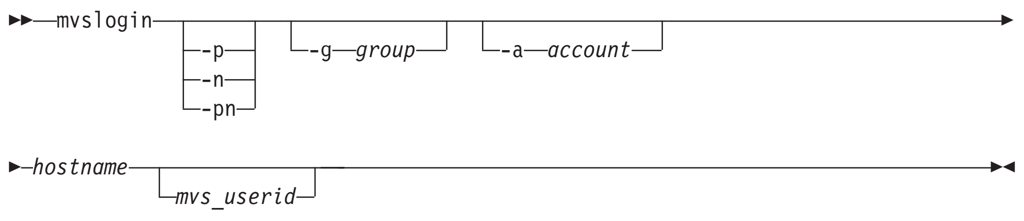
In this example, `smith` is the MVS user ID, `z:` is the drive letter, `mvshost1` is the name of the MVS system, and `test` is the high-level qualifier of the MVS files being accessed. The system responds to this **mount** command by asking for the MVS password.

umount Command and Timing Out

The MVS user ID might be logged off due to a Network File System timeout, when the number of seconds specified in the **logout** attribute is reached, or the server might shut down and start up again. If your MVS user ID, which was authenticated by a PCNFSD request via a **mount** command, is logged off, you need to issue the **umount** and **mount** commands to log on again, because the mount point still exists.

Command Summary for Sun PC-NFS Users

You can use these commands with Sun PC-NFS:



-p Causes a prompt for the user's MVS password. The password is passed to MVS to validate the user logging in. The type of security procedures required by the installation determines whether this parameter is required.

-n Causes a prompt for a new password. The type of security procedures required by the installation determines whether this parameter is used.

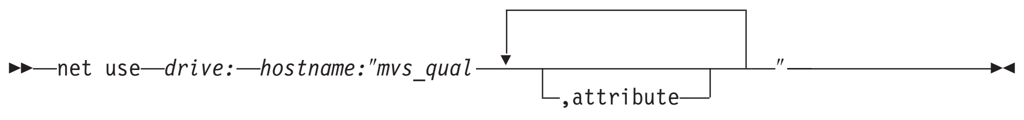
-pn
Causes a prompt for the user's current password and then causes two prompts for the user' new password. The type of security procedures required by the installation determine whether this parameter is used.

-g *group*
A group name string passed to MVS for accounting purposes. The maximum length is 8 characters.

-a *account*
An account string passed to MVS for accounting purposes. The maximum length is 16 characters.

hostname
The name of the MVS host (for example, `mvshost1`).

mvs_userid
A user ID that MVS recognizes as valid. If you do not specify this parameter, your workstation user name is used. The OS/390 NFS server does not support the use of an alias user ID or a mixed case user ID with the **mvslogin** command.



drive
Specifies the drive letter to which the mounted file system is attached.

hostname
The name of the MVS host (for example, `mvshost1`).

mvs_qual
The name of the high-level qualifier of the MVS files that you are accessing in the MVS file system (such as `mvsuser1` or `mvsuser1.test`). The high-level qualifier specified must already exist in the MVS catalog.

attribute
A OS/390 Network File System server data set creation or processing attribute (such as `text`). See "Chapter 10. Descriptions of Attributes for the OS/390 NFS Server" on page 89. If you specify any attributes, make sure you enclose *mvs_qual* and the attributes in double quotation marks.

```
►►──showattr──┬──────┬──hostname──┬────────┬──────────────────────────►◄
              └──-t──┘            └─drive:─┘
```

-t Used to specify tersed output.

hostname
The name of the MVS host (for example, `mvshost1`).

drive
Specifies the drive letter to which the mounted file system is attached.

```
►►──net use──drive:──/d──────────────────────────────────────────────►◄
```

drive
Specifies the drive letter to which the mounted file system is attached.

```
►►──mvslogut──hostname───────────────────────────────────────────────►◄
```

hostname
The name of the MVS host (for example, `mvshost1`).

mvslogin Example

Suppose your MVS user ID is `smith`, and the name of the MVS system that you want to log onto is `mvshost1`. You would enter:

```
C> mvslogin mvshost1 smith
```

After you have logged on, you also need to issue a **net use** command before you can access MVS files.

net use Example

Suppose the name of the MVS system that you want to log onto is `mvshost1`, and the high-level qualifier of the files that you want to access or create is `test`. If you are using PC-NFS and you want to use your `Z:` drive to mount the MVS files, you would enter:

```
C> net use z: mvshost1:test
```

After you have mounted an MVS file or directory, use standard DOS commands to access or create MVS files as if they were local files.

Suppose the default for the processing mode is binary, but you want to mount the `test` files using the text processing mode. You can do this with PC-NFS by entering:

```
C> net use z: mvshost1:"test,text"
```

showattr Example

Suppose you want to see the values of the file attributes associated with the OS/390 Network File System server. To do this, you could enter:

```
C> showattr mvshost1 z:
```

where `mvshost1` is the name of the MVS system, and `Z:` is the drive that you used to mount the MVS files. The data set creation attributes, processing attributes, and site attributes are all displayed. You can modify the data set creation attributes and the processing attributes using the **net use** command, but only the MVS system administrator can change the values of the site attributes.

net use /d Example

Suppose you are finished working with a set of MVS files that you had mounted to your `Z:` drive, and you are ready to unmount them. You would enter:

```
C> net use z: /d
```

mvslogut Example

Use the **mvslogut** command when you finish accessing files that reside on a specific MVS system or host. Invoking the **mvslogut** command prevents other users on your workstation from having access to files on that specific MVS system. You should issue the **mvslogut** command at the close of each session.

For example, if the name of the MVS system that you had logged onto is `mvshost1`, you would log off the MVS system by entering:

```
C> mvslogut mvshost1
```

Chapter 9. Commands and Examples for OS/390 NFS Clients

This chapter gives the syntax and examples of commands that OS/390 NFS users need to know to access MVS, AIX, UNIX, OS/2, OS/390, and other remote files using the OS/390 NFS client. This chapter shows how to:

- Log on to a remote MVS system from the OS/390 NFS client if the target server is a remote OS/390 NFS server.
- Access NFS files from the OS/390 NFS client
- Display OS/390 NFS client statistical data
- Query mount points
- Display default mount point attributes
- Mount and unmount remote file systems from the OS/390 NFS client
- Log out of MVS, if the target server was a remote OS/390 NFS server.

The command programs are intended to run in an OS/390 shell environment and are not implemented as TSO commands, with the exception of **MOUNT** and **UNMOUNT**.

The **MOUNT** and **UNMOUNT** commands are not part of OS/390 NFS. Refer to OS/390 UNIX System Services Command Reference. for additional details. An example of the syntax and usage is shown here for your information. You can use the **TSO HELP MOUNT** and **TSO HELP UNMOUNT** commands to see the syntax that is applicable to your system.

Using Commands on DFSMS/MVS

Here is a summary of the syntax for the commands described in this chapter (the **MOUNT** and **UNMOUNT** commands are discussed later in this chapter):

```
►►──mvslogin──┬──────────────────┬──┬──────────────┬──┬────────────────┬──────────►
              ├─-p───────────────┤  └─-g──group────┘  └─-a──account────┘
              ├─-n───────────────┤
              ├─-pn──────────────┤
              └─-P──mvs_passwd───┘

►─hostname──┬────────────┬──────────────────────────────────────────────────────►◄
            └─mvs_userid─┘
```

-p Causes a prompt for the user's MVS password. The password is passed to MVS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n Causes a prompt for a new password.

-pn
Causes a prompt for the user's current password and then causes two prompts for the user's new password.

-P *mvs_passwd*
No prompt for your MVS password; just type your MVS password after the -P. This enables you to automate your MVS login.

-g *group*
A group name string passed to MVS for accounting purposes. The maximum length is 8 characters.

-a *account*
An account string passed to MVS for accounting purposes. The maximum length is 16 characters.

hostname
The name of the MVS host (for example, `mvshost1`). The default is the local host.

mvs_userid
A user ID that MVS recognizes as valid. If you do not specify this parameter, your workstation user name is used. The OS/390 NFS server does not support the use of an alias user ID or a mixed case user ID with the **mvslogin** command.

```
►►──showattr──┬────┬──hostname──┬────────────┬──────────────────────────►◄
              └─-t─┘            └─/localpath─┘
```

-t Used to specify tersed output.

hostname
The name of the MVS host (for example, `mvshost1`). The default is the local host.

/localpath
The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

```
►►──nfsstat──┬───────────────────┬──────────────────────────────────────►◄
             ├─-c────────────────┤
             ├─-n────────────────┤
             ├─-r────────────────┤
             ├─-z────────────────┤
             └─-m──┬─────────────┤
                   └─mount point─┘
```

-c Displays both NFS and RPC statistics about the OS/390 NFS client. This is the default option on the **nfsstat** command.

-n Displays NFS statistics about the OS/390 NFS client.

-r Displays RPC statistics about the OS/390 NFS client.

-z Initializes statistics to zero. Used by root user only. This option can be combined with options **-c**, **-n**, and **-r** on the **nfsstat** command. When combined with these nfsstat options, each particular set of statistics is set to zero after the statistics are printed.

-m Displays the name of each NFS mounted file system.

-m *mount point*
Displays information for the NFS mounted file system on the specified mount point.

```
►►──showmount──┬────┬──hostname────────────────────────────────────────►◄
               ├─-a─┤
               ├─-d─┤
               └─-e─┘
```

If you omit the options, the default option displays hostnames of all remote mounts from the *hostname* NFS server.

-a Displays all mounts in the format hostname:directory from the hostname specified in the **showmount** command.

-d Displays only directory names of all mounts from the hostname specified in the **showmount** command.

-e Displays the list of exported directories from the hostname specified in the **showmount** command.

hostname
The name of the NFS server host (for example, `mvshost1`). The default is the local host.

```
►►──os22mvs──input──output─────────────────────────────────────────────►◄
```

input
Absolute path name of the input file to be converted.

output
Absolute path name of the output file.

```
►►──mvs2os2──input──output─────────────────────────────────────────────►◄
```

input
Absolute path name of the input file to be converted.

output
Absolute path name of the output file.

```
►►──mvslogout──hostname────────────────────────────────────────────────►◄
```

hostname
The name of the MVS host (for example, `mvshost1`). The default is the local host.

Accessing MVS

To access remote MVS files, enter both the **mvslogin** command to log into the OS/390 NFS server's host system and the **MOUNT** command to mount the files or data sets to your local system. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to saf or safexp. Once the files or data sets are mounted to a local mount point, you can read, write, create, delete and treat the mounted files as part of your local OS/390 UNIX files. When you are finished with your work, use the **UNMOUNT** and

mvslogout commands to break the connection. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

Note: Issuing the **MOUNT** and **UNMOUNT** commands, as well as creation of the mount point, must be performed by someone with superuser authority.

To access files on MVS systems where the OS/390 NFS server site *security* attribute is set to saf, exports, or safexp, you need an MVS user ID and password, and authorization to access the files that you need. You can only establish one MVS session for each MVS user ID. If you do not already have an MVS user ID, an MVS password, and access authorization on the MVS system from which you require NFS services, request them from your MVS system administrator.

Note: If you cannot use the **mvslogin**, **mvslogout**, or **showattr** commands, they might be installed incorrectly. Ensure that your system administrator has made the executable code for these three commands available to your MVS user ID and that you have been given the correct access authority to them.

mvslogin Command Examples

Use the **mvslogin** command to log in to the remote MVS system. The **mvslogin** command can be issued multiple times and the last one overrides the previous one. The **mvslogin** command is only required when accessing data on systems where the OS/390 NFS server site *security* attribute is set to saf or safexp.

Table 14 shows examples of the **mvslogin** command where `mvshost1` is the name of the MVS host and `smith` is the user's ID on MVS.

Table 14. Examples of the mvslogin Command for Clients

Command Examples
`mvslogin -p mvshost1 smith`
`mvslogin -P smithspw -g finance -a 5278 mvshost1 smith`
`mvslogin -n mvshost1 smith`
`mvslogin -pn -a 5278 mvshost1 smith`
`mvslogin mvshost1`
`mvslogin mvshost1 smith`

In the example where the user enters **mvslogin mvshost1**, the current login client user ID is used as the MVS user ID.

In the example where the user enters **mvslogin mvshost1 smith**, the system then prompts for Smith's MVS password. If Smith logs in successfully, this message appears:

```
GFSA955I smith logged in ok.
```

Otherwise, an appropriate error message appears.

Note: Messages that start with GFSA and GFSC apply to Network File System requests. These messages are further explained in "Appendix A. Messages to the Client" on page 103.

“Permission denied” Message

If you have successfully logged in and get the “Permission denied” message while trying to access data, that can be due to one of these cases:

- An **mvslogout** command for the same MVS host has been entered from the same client platform. See **mvslogout** for details.
- Your MVS user ID has been automatically logged out because the **logout** attribute value has been exceeded. This can happen when you leave the client idle for too long. Enter the **mvslogin** command again, and start your processes again. To find out how many seconds you can stay logged in while your client is idle, issue the **showattr** command and look at the **logout** attribute.
- Another mvslogin to the same MVS host using the same MVS ID has been entered from the same UID and the same client platform. If this is the case, retry your access.

MOUNT Command Syntax and Examples

Use the TSO **MOUNT** command to make a connection between a mount point on your local HFS file system and one or more files on a remote MVS, AIX, UNIX, OS/2, OS/390, or other file system. The **MOUNT** command can only be used by a MVS superuser.

Note: The same mount function can also be performed using the OS/390 UNIX **automount** facility or **/etc/rc** shell scripts support. OS/390 UNIX does not support NFS mounts in the SYS1.PARMLIB member statement. When the **automount** facility is used to manage remote NFS mount points, the OS/390 NFS client user could experience ESTALE/EIO errors if the automounter unmounts the accessed mount point when the time limits specified by the **automount** DURATION and DELAY parameters have been exceeded. The **automount** default, DURATION=NOLIMIT, disables the DURATION timeout period. The DELAY for unmounting file systems should be longer than the time OS/390 NFS clients are likely to keep NFS mounts to the files and directories active. For additional information refer to *OS/390 UNIX System Services File System Interface Reference* .

Figure 9 illustrates the syntax of the **MOUNT** command:

```
MOUNT  FILESYSTEM(file_system_name)
       TYPE(NFS)
       MOUNTPOINT(local_mountpoint)
       MODE(RDWR|READ)
       PARM('hostname:"path_name,server_attributes",  options')
       SETUID|NOSETUID
       WAIT|NOWAIT
```

Figure 9. TSO MOUNT Command Syntax OPERANDS

FILESYSTEM(file_system_name)
: Specifies the name of the file system to be added to the file system hierarchy. This operand is required. The file system name specified must be unique among previously mounted file systems. It may be an arbitrary name up to 44 characters in length of a filesystem. You can enclose file_system_name in single quotes, but they are not required.

TYPE(NFS)
Specifies the type of file system that performs the logical mount request. The NFS parameter must be used.

MOUNTPOINT(local_mountpoint)
Specifies the path name of the mount point directory, the place within the file hierarchy where the file system is to be mounted. The local_mountpoint specifies the mount point path name. The name can be a relative path name or an absolute path name. The relative path name is relative to the current working directory of the TSO session (usually the HOME directory). Therefore, you should usually specify an absolute path name. A path name is case-sensitive, so enter it exactly as it is to appear. Enclose the path name in single quotes.

Note: The mount point must be a directory. Any files in that directory are inaccessible while the file system is mounted. Only one file system can be mounted to a mount point at any time.

MODE(RDWR|READ)
Specifies the type of access for which the file system is to be opened.

RDWR specifies that the file system is to be mounted for read and write access. This is the default option.

READ specifies that the file system is to be mounted for read-only access.

PARM('hostname:″path_name,server_attributes″, options')
Specifies the hostname of the remote Network File System server, the server attributes, and the options. The double quotes are omitted if no server attributes are specified. Enclose the entire string in single quotes. You can specify lowercase or uppercase characters.

SETUID|NOSETUID
Refer to OS/390 UNIX System Services Command Reference for details on *SETUID* and *NOSETUID*.

WAIT|NOWAIT
Refer to OS/390 UNIX System Services Command Reference for details on *WAIT* and *NOWAIT*.

The following PARM operand options are supported and can be specified in the **MOUNT** command overriding the installation parameter settings:

acdirmax(n)
Specifies the maximum lifetime of cached directory attributes in seconds. The default value is 60. This option is only valid when *AttrCaching=Y*.

acdirmin(n)
Specifies the minimum lifetime of cached directory attributes in seconds. The default value is 30. This option is only valid when *AttrCaching=Y*.

acregmax(n)
Specifies the maximum lifetime of cached file attributes in seconds. The default value is 60. This option is only valid when *AttrCaching=Y*.

acregmin(n)
Specifies the minimum lifetime of cached file attributes in seconds. The default value is 3. This option is only valid when *AttrCaching=Y*.

AttrCaching(Y|N)
Specifies whether to process attributes and data caching. The default value is Y.

If attribute caching is in effect, OS/390 NFS client maintains cache consistency with the copy of the file on the Network File System server by performing the consistency check with the cached file attributes. When a file's data are read, they remain valid on the OS/390 NFS client until the attribute cache is timed out or negated. If **attrcaching(N)** is specified, it will automatically set **datacaching(N).**

cln_ccsid(n)
Specifies the Coded Character Set Identifier (CCSID) for the local mounted filesystem (OS/390 NFS client). The default is 1047 (LATIN OPEN SYSTEM EBCDIC).

DataCaching(Y|N)
Specifies whether to perform data caching. The default value is Y.

The *DataCaching* attribute provides finer granularity in controlling whether file data should be cached by the OS/390 NFS client. By caching the file data, all subsequent references to the cached data is done locally thus avoiding the network overhead. This has additional significance when obtaining data from Network File System server systems which do not use UNIX access permissions for security as there is a potential security exposure allowing unauthorized users to access file data.

Security checking is done on the Network File System server to determine whether the requesting client user is authorized to access the data. On UNIX systems, this is done by validating the client's user ID and group ID against the file's permission codes. If the authorization checking is successful, the file data is returned to the OS/390 NFS client system. Further authorization checking for subsequent access to the cached data or for other client users is done on the OS/390 NFS client system.

For conventional MVS data set access through the OS/390 NFS server, the user is required to present their MVS credentials which are checked by the MVS security system, such as RACF, before file data is returned. Since the MVS system does not maintain UNIX style permission codes for conventional data sets, the OS/390 NFS server returns a code indicating that **anyone** can access the file. This is done since passing any lesser access code to the client would result in the client user not being allowed to use the cached data which was already read. When the file data is cached on the OS/390 NFS client system and another client user on this system attempts to access the same file data, the OS/390 NFS client checks the returned permission codes to validate access. Since the OS/390 NFS server has passed a code which allows **anyone** access to the file, all users on the client system can access the cached data without further restrictions. If data caching is turned off, no client caching takes place and each user must pass the server security check.

Based on the installation time out values, the file data cached by the client is flushed and further attempts to access the file data again requires passing server authorization.

The installation *DataCaching* parameter can be set and it can be overridden for each mount point so that different mount points can be handled as required for the files under that mount point.

If the potential security exposure can not be tolerated for sensitive file data, the *DataCaching(N)* should be used so that no file data is cached by OS/390 NFS client.

DelayWrite(n)
: Specifies the maximum number of disk blocks for delay write. The valid range is 0 to 32 and the default value is 16. The blocksize is 8192. This option is only valid when *DataCaching=Y*.

Delim (binary|cr|crlf|crnl|lf|lfcr|nl)
: Specifies the record delimiter for record access to remote files via BSAM, QSAM, and VSAM ESDS. The default is *binary*.

  binary specifies the data does not have record delimiters. The access method does not add a delimiter for each record on output and treats any delimiters on input as data.

  The following values specify text.

  cr specifies that records are delimited by the EBCDIC carriage return character (x'0D').

  crlf specifies that records are delimited by the EBCDIC carriage return character followed by the EBCDIC line feed character (x'0D25').

  crnl specifies that records are delimited by the EBCDIC carriage return character followed by the EBCDIC new line character (x'0D15').

  lf specifies that records are delimited by the EBCDIC line feed character (x'25').

  lfcr specifies that records are delimited by the EBCDIC line feed character followed by the EBCDIC carriage return character (x'250D').

  nl specifies that records are delimited by the EBCDIC new line character (x'15').

DynamicSizeAdj(Y|N)
: Specifies whether to perform packet size adjustment for RPC timeout. The default value is Y.

hard|soft
: If you specify hard, then continue to retry NFS RPC until the NFS server responds. If soft is specified, an error is returned if the NFS server does not respond. Specify the maximum number of retries with the *retrans* option. The default value is *hard*. This option is valid for all NFS RPCs under the mount point.

ReadAhead(n)
: Specifies the maximum number of disk blocks to read ahead. The valid range is 0 to 16 and the default value is 1. The blocksize is 8192. This option is only valid when *DataCaching=Y*.

retrans(n)
: Sets the number of times to retransmit NFS RPCs. The valid range is 0 to 1000 and the default value is 3. This option is only valid when *soft* is specified.

retry(n)
: Specifies the number of times to retry the mount operation. The valid range is 0 to 20,000 and the default value is 10,000. This option is only valid during **MOUNT** processing.

rsize(n)
: Sets the read buffer size to n bytes. The valid range is 1 to 8192 and the default value is 8192.

srv_ccsid(n)
Specifies the CCSID for the remote mounted filesystem (NFS server). The default is 819 (ISO 8859-1 ASCII).

timeo(n)
Sets the RPC timeout to n tenths of a second. The default value is 7.

vers(x)
Where x is either 2 or 3. The OS/390 NFS client will communicate with the highest protocol level supported by the remote server using the specified protocol level.

wsize(n)
Sets the write buffer size to n bytes. The valid range is 1 to 8192 and the default value is 8192.

xlat(Y|N)
If Y, specifies the data in all the files are text and the OS/390 NFS client will perform data conversion according to the cln_ccsid and srv_ccsid parameters. The default value is N and should be used for binary data.

Data Conversion

The OS/390 NFS client supports data conversion defined by the Character Data Representation Architecture (CDRA) when reading data from a remote NFS server or writing data to a remote NFS server. CDRA defines a method to identify the different representations of character data. The CCSID attribute specifies an identifier registered with the IBM Character Data Representation Architecture (CDRA) of an encoding scheme for coded character set data. The CCSID attribute is a 16-bit number identifying a specific set of encoding scheme identifier, character set identifier(s), code page identifier(s) and additional coding-related required information that uniquely identifies the coded graphic character representation used. For example, if a file has a CCSID of 437, it is in USA ASCII format. If it has a CCSID of 297, it is in the French EBCDIC format. The meaning of each CCSID is defined in the IBM CDRA. See *Character Data Representation Architecture Reference and Registry* and *Character Data Representation Architecture Overview* for additional information.

The *cln_ccsid*, *srv_ccsid*, and *xlat* parameters identify whether data conversion is performed, and how data conversion is done. The *cln_ccsid*, *srv_ccsid*, and *xlat* parameters are supported by the OS/390 NFS client installation parameter and **TSO MOUNT** command. The parameters on a **TSO MOUNT** command override the parameters specified in OS/390 NFS client installation parameter.

Attention:

If *xlat*(Y) is specified to perform ASCII/EBCDIC conversion and the default CCSID character sets are used, any application programs running under the OS/390 shell and using the OS/390 NFS client needs to be aware that the line-feed (LF) delimiter will be translated into newline (NL).

The default OS/390 NFS client *cln_ccsid* (1047) and *srv_ccsid* (819) CCSID character sets will invoke CDRA to use an alternative conversion table which will convert ASCII line-feed (LF) into EBCDIC newline (NL) and convert both EBCDIC line-feed (LF) and newline (NL) into ASCII line-feed (LF).

Data conversion is supported on a mount point level. All the file objects under the same mount point have the same *xlat*, *cln_ccsid*, and *srv_ccsid* values. They cannot be set on a file basis under the mount point.

BSAM, QSAM, and VSAM ESDS Access to Remote Files

BSAM, QSAM, and VSAM ESDS applications can access files stored on remote NFS servers through the OS/390 NFS client. This will allow existing MVS application programs access to data on other systems using BSAM, QSAM, and VSAM ESDS interfaces. The BSAM, QSAM, and VSAM ESDS access methods assume that all text files are EBCDIC. When using these access methods, the *Delim* parameter indicates whether the remote files contain text or binary data. Text data consists of records that are separated by a delimiter. If the *Delim* parameter is not binary, the EBCDIC text delimiter is used by the access methods when processing the remote files. The *Delim* parameter is supported on the OS/390 NFS client installation parameter and **TSO MOUNT** command.

All the remote file objects under the same mount point have the same *Delim* value. The *Delim* parameter cannot be set on a file basis under the mount point. The *Delim* parameter in the **TSO MOUNT** command overrides the *Delim* parameter specified in OS/390 NFS client installation parameter. However, you can override the *Delim* parameter on the **TSO MOUNT** command with the **FILEDATA** parameter on a JCL DD statement, SVC 99, or TSO **ALLOCATE** command. The **FILEDATA** parameter can be either *TEXT* or *BINARY*.

For BSAM, QSAM, and VSAM ESDS applications accessing files stored on remote NFS servers, the OS/390 NFS client provides data conversion when the *xlat=Y* parameter is specified according to the *cln_ccsid* and *srv_ccsid* settings. When *xlat*=N, the OS/390 NFS client will not perform data conversion. The **FILEDATA** parameter on a JCL DD statement is also used to specify if the data consists of text records separated by delimiters or if the data is binary and does not contain record delimiters. To avoid undesirable data conversions, care should be taken to insure the specification of the *xlat* and *Delim* parameters are not in conflict with the data type specified by the **FILEDATA** parameter on a JCL DD statement. The **FILEDATA** and *Delim* parameters only affect BSAM, QSAM, and VSAM ESDS data access and have no affect on the OS/390 NFS client data conversion. The OS/390 NFS client data conversion is only controlled by the *xlat* parameter. The significance of different **FILEDATA** and *Delim* combinations are as follows:

Note: In each case insure the OS/390 NFS client *xlat=Y*, *cln_ccsid*, and *srv_ccsid* parameter settings are correct for the **FILEDATA** and *Delim* combination.

FILEDATA=TEXT, *Delim***=notBINARY**
: Means that the data is to be accessed as text. The access method appends a record delimiter on output and expects delimiters on input. The delimiter used is that specified on the *Delim* parameter.

FILEDATA=TEXT, *Delim***=BINARY**
: Means that the data is to be accessed as text. The access method appends a record delimiter on output and expects delimiters on input. The delimiter used is the default of the EBCDIC new line character (x’15’) since the *Delim* parameter does not specify a valid text delimiter.

FILEDATA=BINARY, *Delim***=notBINARY**
: Means that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

FILEDATA=BINARY, *Delim***=BINARY**
: Means that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

FILEDATA=not specified, ***Delim*****=specified**
: Means that the data is to be accessed according to the value specified in the *Delim* parameter.

FILEDATA=not specified, ***Delim*****=not specified**
: Means that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

The OS/390 NFS client also provides UNIX authentication for security and provides the OS/390 UNIX client's user ID (UID), group ID (GID), and a list of users GIDs to the remote NFS server for authorization checking. When the remote NFS server is the OS/390 NFS server, the **mvslogin** command can be used to provide additional security checking through RACF authentication. MVS application programs which require access to data on remote OS/390 systems may be required to perform an **mvslogin**.

For information on BSAM, QSAM, and VSAM ESDS applications access to HFS or remote files and their restrictions, refer to *DFSMS/MVS Using Data Sets*.

MOUNT Command Examples

Use the **TSO MOUNT** command to make a connection between a local mount point on the OS/390 NFS client and an NFS server.

In this example, the **mount** command is used to mount a set of MVS files. The PARM operand contains the OS/390 NFS server *text* processing attribute and requires the use of double quotes around the string *user,text*.

```
mount filesystem(nfs00) type(nfs) mountpoint('/u/nfsdir')
parm('stlmvs3:"user,text",soft,timeo(100)')
```

nfs00	The name of the file system to be added to the file system hierarchy
nfs	The required file system type
/u/nfsdir	The name of the mount point (preferably an empty directory)
stlmvs3	The name of the host OS/390 NFS server
user	The name of the high-level qualifier of the MVS files on the OS/390 NFS server
text	Processing attribute for the OS/390 NFS server
soft	PARM operand option for the OS/390 NFS client
timeo(100)	PARM operand option for the OS/390 NFS client

In this example, the **mount** command is used to mount a MVS HFS directory. The PARM operand contains the OS/390 NFS server *binary* processing attribute and requires the use of double quotes around the string */hfs/u/user,binary*.

```
mount filesystem(nfs01) type(nfs) mountpoint('/u/nfsdir1')
parm('stlmvs3:"/hfs/u/user,binary",soft')
```

nfs01	The name of the file system to be added to the file system hierarchy
nfs	The required file system type

/u/nfsdir1
The name of the mount point (preferably an empty directory)

stlmvs3
The name of the host OS/390 NFS server

/hfs/u/user
The name of the HFS directory on the OS/390 NFS server

binary Processing attribute for the OS/390 NFS server

soft PARM operand option for the OS/390 NFS client

In this example, the **mount** command is used to mount an AIX home directory.

```
mount filesystem(nfs02) type(nfs) mountpoint('/u/nfsdir2')
parm('aix6000:/home/user,xlat(y)')
```

nfs02 The name of the file system to be added to the file system hierarchy

nfs The required file system type

/u/nfsdir2
The name of the mount point (preferably an empty directory)

aix6000
The name of the host AIX NFS server

/home/user
The name of the home directory on the AIX NFS server

xlat(y) PARM operand option for the OS/390 NFS client

In this example, the **mount** command is used to mount an AIX home directory using the NFS version 3 protocol.

```
mount filesystem(nfs03) type(nfs) mountpoint('/u/nfsdir2')
parm('aix6000:/home/user,xlat(y),vers(3)')
```

nfs03 The name of the file system to be added to the file system hierarchy

nfs The required file system type

/u/nfsdir2
The name of the mount point (preferably an empty directory)

aix6000
The name of the host AIX NFS server

/home/user
The name of the home directory on the AIX NFS server

xlat(y) PARM operand option for the OS/390 NFS client

vers(3)
The version of NFS protocol that is being used

In this example, the **mount** command is used to mount an OS/2 directory.

```
mount filesystem(nfs03) type(nfs) mountpoint ('/u/httpd')
parm('os2_server:d:\httpd')
```

nfs03 The name of the file system to be added to the file system hierarchy

nfs The required file system type

/u/httpd
The name of the mount point (preferably an empty directory)

os2_server
The name of the host OS/2 NFS server

d:\httpd
The name of the directory on the OS/2 NFS server

Getting Authorization to Access Files

If the mount fails, check with your system administrator to ensure that you are authorized to access the HFS, AIX, UNIX, or OS/2 file system listed in the exports control file. The OS/390 NFS client also provides UNIX authentication for security and provides the OS/390 UNIX client's UID, GID, and a list of the GIDs from the OS/390 UNIX client's groups to the remote NFS server for authorization checking. When the remote NFS server is the OS/390 NFS server, the **mvslogin** command can be used to provide additional security checking through RACF authentication. The privilege level required to enter **MOUNT** and **UNMOUNT** commands is superuser. When requesting service from the OS/390 NFS server, if the MVS system operator issues the *freeze=on* operand of the **MODIFY** command, all new tries to mount an HFS filesystem fail until the MVS system operator issues the *freeze=off* operand. If the OS/390 UNIX system operator issues the *freeze=onhfs* operand of the **MODIFY** command, conventional MVS data sets can still be mounted, but all new tries to mount OS/390 UNIX files fail until the system operator issues the *freeze=offhfs* operand.

Saving of Mount Points

Once the **MOUNT** command is issued successfully and a mount point is established between a remote directory and the HFS, the mount point information is not saved by the OS/390 NFS client. The OS/390 NFS client does not maintain mount persistence across restart. If OS/390 UNIX or the OS/390 NFS client is restarted, all prior session's mount point information is lost and all mount points must be reestablished.

Automatic Timed Logout — logout Attribute

When using Network File System services from the OS/390 NFS server, if there is no activity on the client within the period specified in the **logout** attribute of the attributes file, or if the server stops, the connection between the server and the client workstation is logged out automatically. You must issue the **mvslogin** command again to get access to the MVS files.

Converting Line Delimiters Between OS/2 and MVS — os22mvs, mvs2os2

The OS/390 NFS client only supports file data conversion. When working with files accessed via an OS/2 NFS server, the **os22mvs** and the **mvs2os2** command programs should be used to convert the line delimiters from OS/2 format to MVS format and vice versa.

Use the **os22mvs** command to convert the line delimiters in an OS/2 file from OS/2 format to MVS format. The carriage-return line-feed pairs (CR, LF) in the input file are converted to newline (NL) and the end of file character (EOF) is ignored. No other conversions are performed. Both the input and output file can be a local or remote file.

Use the **mvs2os2** command to convert the line delimiters in an MVS file from MVS format to OS/2 format. All newline (NL) characters are converted to carriage-return line-feed pairs (CR, LF). No other conversion is performed. Both the input and output file can be a local or remote file.

Displaying Client Statistical Information — nfsstat

Use the **nfsstat** command to display the OS/390 NFS client statistical information, to reset the statistical information to zero, to display NFS mount point information, or to set the debug status.

The following **nfsstat** command displays the NFS and RPC statistics for the OS/390 NFS client:

```
nfsstat -c
```

The following **nfsstat** command initializes the NFS and RPC statistics for the OS/390 NFS client to zero. This option may be used by the root user only:

```
nfsstat -z
```

Figure 10 shows the output from the **nfsstat** command using the -c option to display the RPC and NFS statistics for the NFS client.

```
USER1:/u/user1:>nfsstat -c

Client rpc:
calls           badcalls        retrans         timeout
51              0               0               0

Client nfs_V2:
calls        badcalls
45           0
null         getattr     setattr         root         lookup
0        0%  21     47%  1          2%   0       0%  13         29%
readlink     read        writecache      write        create
0        0%  0       0%  0          0%   0       0%  1           2%
remove       rename      link            symlink      mkdir
4        9%  0       0%  0          0%   0       0%  0           0%
rmdir        readdir     fsstat
0        0%  3       7%  2          4%


Client nfs_V3:

calls           badcalls
4            0
null         getattr      setattr         lookup       access
0        0%  0        0%  0           0%  0        0%  0        0%
readlink     read         write           create       mkdir
0        0%  0        0%  0           0%  0        0%  0        0%
symlink      mknod        remove          rmdir        rename
0        0%  0        0%  0           0%  0        0%  0        0%
link         readdir      readdirplus     fsstat       fsinfo
0        0%  0        0%  0           0%  2       50%  1       25%
pathconf     commit
1       25%  0        0%
```

Figure 10. Displaying NFS Client RPC and NFS Statistical Information

Figure 11 shows the output from the **nfsstat** command using the -r option to display the RPC statistics for the NFS client.

```
#nfsstat -r
Client rpc:
calls           badcalls        retrans         timeout
61              0               20005           20005
```

Figure 11. Displaying NFS Client RPC Statistical Information

calls Total number of RPC calls sent

badcalls
Total of RPC calls rejected by a server

retrans
Number of times an RPC call had to be retransmitted

timeout
Number of times an RPC call timed out

Figure 12 shows the output from the **nfsstat** command using the -n option to display the NFS statistics for the NFS client.

```
# nfsstat -n

Client nfs_V2:
calls        badcalls
45           0
null         getattr     setattr         root         lookup
0        0%  21     47%  1           2%  0        0%  13         29%
readlink     read        writecache      write        create
0        0%  0       0%  0           0%  0        0%  1           2%
remove       rename      link            symlink      mkdir
4        9%  0       0%  0           0%  0        0%  0           0%
rmdir        readdir     fsstat
0        0%  3       7%  2           4%


Client nfs_V3:
calls           badcalls
4            0
null         getattr      setattr         lookup       access
0        0%  0        0%  0           0%  0        0%  0        0%
readlink     read         write           create       mkdir
0        0%  0        0%  0           0%  0        0%  0        0%
symlink      mknod        remove          rmdir        rename
0        0%  0        0%  0           0%  0        0%  0        0%
link         readdir      readdirplus     fsstat       fsinfo
0        0%  0        0%  0           0%  2       50%  1       25%
pathconf     commit
1       25%  0        0%
```

Figure 12. Displaying NFS Client NFS Statistical Information

calls Total number of NFS calls sent

badcalls
Total of NFS calls rejected by a server

Figure 13 shows the output from the **nfsstat** command using the -m option to display the server and path name of each NFS mounted file system.

```
#nfsstat -m
mvshost:/sj/sjpl is mounted on /sj/sjpl/host1
        filesystem NFS_MNT1
mvshost2:/sj/sjpl2,text mounted on /sj/sjpl/host2
        filesystem NFS_MNT2
```

Figure 13. Displaying NFS Mounted File System Information

Figure 14 and Figure 15 show the output from the **nfsstat** command using the -m option to display the server name, path name, and attributes of mount point */sj/sjpl/host1* from both version 2 and version 3 protocols.

```
#nfsstat -m /sj/sjpl/host1
server  mvshost
path    /sj/sjpl/host1

hard    vers(2)   retry(10)   timeo(7)        retrans(3)
delim(binary)   xlat(n)       cln_ccsid(1047)  srvccsid(819)
rsize(8192)     wsize(8192)   readahead(8)    delaywrite(16)
acregmin(3)     acregmax(60)  acdirmin(30)    acdirmax(60)
datacaching(y)  attrcaching(y)  dynamicsizeadj(y)
```

Figure 14. Displaying V.2 Protocol NFS Mounted File System Information

```
#nfsstat -m /sj/sjpl/host1
server  mvshost
path    /sj/sjpl/host1

hard    vers(3)   retry(10)   timeo(7)        retrans(3)
delim(binary)   xlat(n)       cln_ccsid(1047)  srvccsid(819)
rsize(32768)     wsize(32768)   readahead(8)    delaywrite(16)
acregmin(3)     acregmax(60)  acdirmin(30)    acdirmax(60)
datacaching(y)  attrcaching(y)  dynamicsizeadj(y)
```

Figure 15. Displaying V.3 Protocol NFS Mounted File System Information

Displaying Server Mount Information — showmount

Use the **showmount** command to display the remote NFS server mount information. If you omit the options, the default option displays hostnames of all remote mounts from the *hostname* NFS server. If you omit the *hostname* parameter, then the local hostname is used.

The following **showmount** command displays all remote mounts in the format hostname:directory from the local *hostname* NFS server:

```
showmount -a
```

The following **showmount** command displays only the directory names of all the remote mounts from the local *hostname* NFS server:

```
showmount -d
```

Figure 16 shows the output from the **showmount** command using the -a option to display all mounts in the format hostname:directory from the hostname *mvshost*.

```
# showmount -a mvshost
mvshost.sanjose.ibm.com:/IBMUSER
usera.sanjose.ibm.com:/USER2
```

Figure 16. Displaying Server Mount Hostname and Directory Information

Figure 17 shows the output from the **showmount** command using the -d option to display only the directory names of all mounts from the hostname *mvshost*.

```
# showmount -d mvshost
/IBMUSER
/USER2
```

Figure 17. Displaying Server Mount Directory Information

Figure 18 shows the output from the **showmount** command with no option specified to display only the hostnames of all remote mounts from the hostname *mvshost*.

```
# showmount mvshost
mvshost.sanjose.ibm.com
usera.sanjose.ibm.com
```

Figure 18. Displaying Server Mount Hostname Information

Figure 19 shows the output from the **showmount** command using the -e option to display the exported directories from the hostname *aix_server1*.

```
USER1:/u/user1:>showmount -e aix_server1
Export list for host aix_server1:
/home/u/guest/test (everyone)
/usr/lpp/info      (everyone)
/tmp               (everyone)
```

Figure 19. Displaying Server Export Information

Figure 20 on page 84 shows the output from the **showmount** command using the -e option to display the exported directories from the hostname *mvshost*. In this case, *mvshost* has the site attribute set to **security(none)**.

```
# showmount -e mvshost
No exported file systems for host MVSHOST
```

Figure 20. Displaying Server Export Information, security(none)

Figure 21 shows the output from the **showmount** command using the -e option to display the exported directories from the hostname *mvshost*. In this case, *mvshost* has the site attribute set to **security(safexp)**.

```
# showmount -e mvshost
Export list for host MVSHOST:
/IBMUSER              user1
```

Figure 21. Displaying Server Export Information, security(safexp)

Displaying Default and Mount Point Attributes — showattr

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point of the OS/390 NFS server. If you specify a mount point, **showattr** shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see “Chapter 10. Descriptions of Attributes for the OS/390 NFS Server” on page 89. and “MOUNT Command Syntax and Examples” on page 71.

If you omit the *hostname*, you must specify the */localpath*.

The following is an example of the **showattr** command:

```
showattr mvshost1 /u/smith/mnt
```

Make sure that your version of the showattr command matches the release of the OS/390 NFS that you are using. Otherwise, the OS/390 NFS server attributes will not display.

These examples show different ways you can use the **showattr** command.

Figure 22 on page 85 shows a **showattr** command with just the hostname (`mvshost1` in this example) specified. The default attributes for the server are displayed.

```
USER1:/u/user1:>showattr mvshost1

OS/390 Network File System Server Data Set Creation Attributes:

lrecl(8196)            recfm(vb)              blksize(0)
space(100,10)          blks                   dsorg(ps)
dir(27)                unit()                 volume()
recordsize(512,4K)     keys(64,0)             nonspanned
shareoptions(1,3)
mgmtclas()             dsntype(pds)           norlse
dataclas()             storclas()

OS/390 Network File System Server Processing Attributes:

binary                 lf                     blankstrip
nofastfilesize         noretrieve             maplower
mapleaddot             executebitoff          setownerroot
attrtimeout(120)       readtimeout(90)        writetimeout(30)
sync                   nofileextmap           xlat(OEMVS)
sidefile(hlq.nfs.mapping)

OS/390 Network File System Server Site Attributes (not
modifiable):

mintimeout(1)          nomaxtimeout           logout(1800)
nfstasks(8,8)          restimeout(48,0)       hfs(/hfs)
bufhigh(32M)           readaheadmax(16K)      cachewindow(128)
percentsteal(20)       maxrdforszleft(32)     logicalcache(16M)
smf(userfile)          nopcnfsd               security(exports,exports,exports)
leadswitch             sfmax(0)               checklist
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Figure 22. Displaying Default Attributes

If you use the terse (-t) option, the attributes display like this:

```
USER1:/u/user1:>showattr -t mvshost1

lrecl(8196),recfm(vb),blksize(0),space(100,10),blks,dsorg(ps),dir(27),unit(),
volume(),recordsize(512,4K),keys(64,0),nonspanned,shareoptions(1,3),mgmtclas(),
dsntype(pds),norlse,dataclas(),storclas()
binary,lf,blankstrip,nofastfilesize,retrieve,maplower,mapleaddot,executebitoff,
setownerroot,attrtimeout(120),readtimeout(90),writetimeout(30),sync,
nofileextmap,xlat(oemvs),sidefile(hlq.nfs.mapping)
mintimeout(1),nomaxtimeout,logout(1800),nfstasks(8,8),restimeout(48,0),
hfs(/hfs),bufhigh(32M),readaheadmax(16K),cachewindow(128),percentsteal(20),
maxrdforszleft(32),logicalcache(16M),smf(userfile),nopcnfsd,
security(exports,exports,exports),leadswitch,sfmax(0),checklist,
public(NFSUSR3,/HFS/usr/lpp/internet/server_root/pub)
```

Unmounting and Logging Out of MVS

This section describes the **UNMOUNT** and **mvslogout** commands.

Disconnecting Your Mount Point — UNMOUNT

Use the **UNMOUNT** command to break the connection between the mount point on your client and the server (that is, to unmount). You must have superuser authority to issue the UNMOUNT command.

Note: The same unmount function can also be performed using the OS/390 UNIX **automount** facility or /etc/rc shell scripts support. When the **automount** facility is used to manage remote NFS mount points, the OS/390 NFS client user could experience ESTALE/EIO errors if the automounter unmounts the accessed mount point when the time limits specified by the **automount**

DURATION and DELAY parameters have been exceeded. For additional information on the OS/390 UNIX **automount** facility (**/etc/rc** shell scripts support) refer to *OS/390 UNIX System Services Planning* and*OS/390 UNIX System Services File System Interface Reference*.

Figure 23 illustrates the syntax of the **UNMOUNT** command:

```
UNMOUNT   FILESYSTEM(file_system_name)
           NORMAL | DRAIN | IMMEDIATE | FORCE | RESET
```

Figure 23. TSO UNMOUNT Command Syntax OPERANDS

FILESYSTEM(file_system_name)
Specifies the name of the file system to be removed from the file system hierarchy. file_system_name specifies the file_system_name exactly as it was specified when the file system was originally mounted. You can enclose file_system_name in single quotes, but they are not required.

NORMAL|DRAIN|IMMEDIATE|FORCE|RESET
NORMAL: Specifies that if no user is accessing any of the files in the specified file system, the unmount request is processed. Otherwise, the system rejects the unmount request. *NORMAL* is the default option.

DRAIN: Specifies that the system is to wait until all uses of the file system have ended normally before the unmount request is processed or until another UNMOUNT command is issued.

Note: **UNMOUNT** can be specified with *IMMEDIATE* to override a previous **UNMOUNT** *DRAIN* request for a file system. If this is used in the foreground, your TSO session waits until the **UNMOUNT** request has completed. The attention request key (usually ATTN or PA1) will not end the command.

IMMEDIATE: Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified system receive failing return codes. All data changes to files in the specified file system are saved. If the data changes to files cannot be saved, the unmount request fails.

FORCE: Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified file system receive failing return codes. All data changes to files in the specified file are saved, if possible. If the data changes cannot be saved to the files, the unmount request continues and data is lost.

Note: You must issue an **UNMOUNT** *IMMEDIATE* request before issuing **UNMOUNT** *FORCE*. Otherwise, **UNMOUNT** *FORCE* fails.

RESET: A reset request stops a previous **UNMOUNT** *DRAIN* request.

The following example unmounts the file system NFSC_001 normally.

```
UNMOUNT FILESYSTEM('NFSC_001')
```

The following example forces an unmount of the file system NFSC_001. You must issue an **UNMOUNT** *IMMEDIATE* before you can issue an **UNMOUNT** *FORCE* command.

```
UNMOUNT FILESYSTEM('NFSC_001') IMMEDIATE
UNMOUNT FILESYSTEM('NFSC_001') FORCE
```

If you receive a **“No Such File or Directory”** message, the MVS system operator can also unmount your workstation from the server. If this happens before you try to unmount, you get a “No such file or directory” error message.

Ending Your MVS Session — mvslogout

Use the **mvslogout** command to disconnect from the remote OS/390 NFS server host. The **mvslogout** command is only required when the **mvslogin** command was used to begin the connection.

An **mvslogout** to an MVS user ID cancels a prior **mvslogin** to the same MVS user ID from the same local host.

Your account is automatically logged out if it is inactive for the period of time specified in the **logout** site attribute.

The following example disconnects the client from the remote OS/390 NFS server machine, **mvshost1**:

```
mvslogout mvshost1
```

Chapter 10. Descriptions of Attributes for the OS/390 NFS Server

This chapter contains tables describing the attributes used to manipulate files in the OS/390 NFS server. This chapter contains descriptions for:

- Data set creation attributes
- Processing attributes
- Site attributes

There are two kinds of attributes the client user can modify:

- Data set creation attributes provide information about an MVS file to the OS/390 NFS server, such as the type of file, or how the file is allocated (for example, blocks, cylinders, or tracks).
- Processing attributes provide information to the OS/390 NFS server about how to handle the file, such as how long the files remain open, or whether the files are processed in text or binary format.

Site attributes can only be modified by the system administrator.

Note: For information on attributes for the OS/390 NFS client, refer to the chapter on customizing the Network File System in *OS/390 Network File System Customization and Operation* and the chapter on commands and examples for the OS/390 NFS client in *OS/390 Network File System User's Guide*.

Attributes Used for OS/390 UNIX System Services File Access

These attributes are specific to OS/390 UNIX file access:

- **hfs***(prefix)*
- **sync** and **async**
- **extlink**

These attributes are relevant to accessing OS/390 UNIX files as well as conventional MVS data sets:

logout	User log time out
security	Security checking
text	ASCII to EBCDIC data conversion and vice versa
binary	No ASCII and EBCDIC data conversion
xlat	Customized translation table

Using Multipliers

Instead of entering the entire numeric values for the attributes, you can use the multipliers K (1024), M (1024 × 1024), or G (1024 × 1024 × 1024). For example, entering `10M` is the same as entering `10,485,760`.

Specifying Attributes Multiple Times

Specifying an attribute several times on a line does not cause an error. The line is read from left to right, and the last of any duplicate attribute is used. For example:

```
$ vi "file,recfm(vb),recfm(fb)"
```

This results in a file created with a fixed-blocked format.

Data Set Creation Attributes

The data set creation attributes are used to define the structure of MVS data sets when creating a file. These attributes correspond to the data control block (DCB) or the job control language (JCL) parameters used to define an MVS data set when it is created. Refer to *OS/390 MVS JCL Reference* for more detailed information about the data set creation attributes.

The data set creation attributes are described in Table 15. The data set creation attributes do not apply for HFS data sets. Defaults are underlined **in this format**.

You can override these attributes by using the **mount** command or file creation command. For PDS and PDSE, members have the same attributes as the data set attributes, so the file creation attributes specified for members are ignored.

Table 15. Data Set Creation Attributes

Data Set Creation Attribute	Description
blks	Specifies that disk space (see the **space** attribute in this table) is allocated by blocks, except for VSAM data sets.
cyls	Specifies that disk space is allocated by cylinders.
recs	Specifies that disk space is allocated by records for VSAM data sets. **blks** and **recs** are identical for VSAM data sets.
trks	Specifies that disk space is allocated by tracks.
blksize(0 | *quan*)	Specifies the maximum length, in bytes, of a physical block on disk. *quan* is a number from **0** (the default) to 32,760. If blksize(0) is specified, the system determines an optimal block size to use.
dataclas(*class_name*)	Specifies the data class associated with the file creation. The *class_name* must be defined to DFSMS/MVS before it can be used by the client. The system-managed storage automatic class selection routine must also assign a storage class to the file being created. For more information on data classes, refer to *DFSMS/MVS DFSMSdfp Storage Administration Reference*.
dir(27 | *quan*)	Specifies the number of 256-byte records needed in the directory of a PDS. Use it with the **mkdir** command when you are creating a PDS. *quan* is a number from 1 to 16,777,215 (the default is **27**). The maximum number of PDS members is 14,562.
dsntype(library | pds)	Specifies whether a PDSE or a PDS is to be created when the **mkdir** client command is used. **library** is for PDSE. **pds** is for PDS. You cannot create a PDS (or PDSE) within another PDS (or PDSE). If you need help deciding whether to create a PDS or a PDSE, refer to *DFSMS/MVS Using Data Sets*.

Table 15. Data Set Creation Attributes (continued)

Data Set Creation Attribute	Description
dsorg(*org***)**	Specifies the organization of a data set. *org* can be a physical sequential (**ps**) data set, direct access (**DA**) data set, VSAM KSDS (**indexed**), VSAM RRDS (**numbered**) or VSAM ESDS (**nonindexed**). This attribute is ignored for directory-oriented client commands. If you are using VSAM data sets in binary mode with an AIX client, then **nonindexed** is recommended.
keys(*len, off***)**	Specifies the length and offset of the keys for VSAM KSDS data sets. Keys can only be specified when using **dsorg(indexed)**. *len* and *off* are specified in bytes.*len* is between 1 and 255 (the default is **64**). *off* is between 0 and 32,760 (the default is **0**). When you create a VSAM KSDS data set, the records you are loading into it must be keyed-sequenced or the write fails. Each write of the data set is treated like a first load, and requires that the records being loaded are in ascending key sequence.
lrecl(8196 | ***quan*)**	Specifies: 1. The length, in bytes, for fixed-length records. 2. The maximum length, in bytes, for variable-length records. If the **blksize** attribute is specified, the value must be at least 4 bytes less than the **blksize** quantity. *quan* is a number from 1 to 32,760 (the default is **8196**).
mgmtclas(*mgmt_class_name***)**	Specifies the management class associated with the file creation. The *mgmt_class_name* must be defined to DFSMS/MVS before it can be used by the client. The system managed storage automatic class selection (ACS) routine must also assign a storage class to the file being created. For more information on management classes, see*DFSMS/MVS DFSMSdfp Storage Administration Reference*.
recfm(*cccc***)**	Specifies the format and characteristics of the records in the data set. *cccc* can be 1 to 4 characters, in one of the following combinations: `f | fb | fs | fbs` `u` `v | `**`vb`**` | vs | vbs` Valid record format characters: *b* Blocked; *f* Fixed-length records; *s* Spanned for variable records, standard format for fixed records; *u* Undefined-length records; *v* Variable-length records. In **recfm**, codes **v**, **f** and **u** are mutually exclusive. The **s** code is not allowed for a PDS or PDSE.

Table 15. Data Set Creation Attributes (continued)

Data Set Creation Attribute	Description
recordsize(*avg,max***)**	The average and maximum record size for VSAM data sets. *avg* and *max* are specified in bytes. They can each range from 1 to 32,760 (the defaults are **512** and **4096**, respectively). **These values must be equal for VSAM RRDS.**
rlse	Specifies that unused space should be released from the data set the first time a new data set is closed. For slow clients with long pauses between writes, the **rlse** attribute causes space to be released from the primary extent prematurely. Further writes cause secondary space to be allocated.
norlse	Specifies that unused space should not be released from the data set.
shareoptions(*xreg,xsys***)**	Specifies the cross-region and cross-system share options for a VSAM data set. *xreg* is a number from 1 to 4; *xsys* is either 3 or 4. The defaults are **1** and **3**, respectively.
	This applies to VSAM data sets only. For spanned records of non-VSAM data sets, see the entry for **recfm** in this table.
spanned	Specifies that VSAM KSDS or ESDS data sets can contain records that span control intervals (spanned records).
nonspanned	Specifies that data sets do not have spanned records.
space(*prim[,aux]***)**	Specifies the amount of primary and auxiliary space allocated for a new data set on a direct access volume. *prim* is the number (from 0 to 16,777,215) of primary tracks, cylinders, or data blocks in the data set. *aux* (optional) is the number (from 0 to 16,777,215) of additional tracks, cylinders, or blocks allocated if more space is needed. If this attribute is not specified, the default is used. The defaults are **100** and **10**, respectively.
storclas(*class_name***)**	Specifies the storage class associated with the file creation. The *class_name* must be defined to the DFSMS/MVS before it can be used by the client. For more information on storage classes, refer to *DFSMS/MVS DFSMSdfp Storage Administration Reference*.
unit(*unit_name***)**	Specifies the unit on which to create a data set. *unit_name* is a generic or symbolic name of a group of DASD devices. The *unit_name* must be specified as 3390 for extended format data sets. **Note:** You cannot create or access tape data sets on an MVS host using the OS/390 NFS server. You cannot create extended format data sets with the OS/390 NFS server, except via ACS routines.
vol(*volser***)**	Specifies the name of the DASD volume to be used to store the created data set. **vol** is the keyword and *volser* represents the volume name. If a data set is to be system-managed, as is determined by the DFSMS/MVS Automatic Class Selection (ACS) routines, you can omit this attribute.

Processing Attributes

Processing attributes are used to control how files are accessed by the client. The processing attributes are described inTable 16. Defaults are underlined **in this format**. You can override the default processing attributes.

Table 16. Processing Attributes

Processing Attribute	Description
attrtimeout(*n*)	The time (in seconds) that the data set remains allocated after a **lookup** or **getattr** server operation. The default is **120**. *n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). **Note: lookup** is an NFS protocol that searches for a file in the current directory. If it finds the file, **lookup** returns information on the file's attributes and a file handle pointing to the file. The **attrtimeout** value is normally greater than the **readtimeout** or **writetimeout**values.
noattrtimeout	The data set is not deallocated after a **lookup** or **getattr** operation. For more information, see "Timeout Attributes" on page 96, following this table.
binary	Indicates that the data set is processed between the client and server using binary format and no data conversion occurs between ASCII and EBCDIC formats.
text	Converts the contents in the data set between EBCDIC and ASCII formats. Use this format to share text data between clients and MVS applications. In text mode, the following attributes apply only to conventional MVS data sets: • **blankstrip** and **noblankstrip**. See the entry for **blankstrip** in this table. • End-of-line specifiers (**lf**, **cr**, **lfcr**, **crlf**, or **noeol**) are used to indicate the MVS logical record boundary. See the entry for **lf** in this table. See the **xlat** attribute in this table for customized EBCDIC-ASCII tables.
blankstrip	With text mode, strips trailing blanks at the end of each record of a fixed-length text file when the file is read. Pads the end of each file or record with blanks when a text file is written.
noblankstrip	Does not strip trailing blanks at the end of fixed-length records when a fixed-length text file is read. Does not pad records when writing a text file. The file must be of the correct size or an I/O error is reported to the client. For information on the **text** attribute, see the entry for **binary** in this table. This attribute does not apply to HFS data sets.
	With text mode, use one of the following end-of-line specifiers:
lf	Line Feed is the end-of-line terminator (standard AIX or UNIX).
cr	Carriage Return is the end-of-line terminator.
lfcr	Line Feed followed by Carriage Return is the end-of-line terminator.
crlf	Carriage Return followed by Line Feed is the end-of-line terminator (standard DOS).
noeol	No end-of-line terminator. For information on the **text** attribute, see the entry for **binary** in this table. This attribute does not apply to HFS data sets.

Table 16. Processing Attributes (continued)

Processing Attribute	Description
executebiton	Turns on the execute bits in user, group, and other (as reported with the **ls** (list) AIX or UNIX command) for a mount point's files. Use when storing executable or shell scripts on the MVS system. This option can only be overridden on a mount point basis — not at a command level. This attribute does not apply to HFS files and can only be used with the mount command.
executebitoff	Turns off the execute bits in user, group, and other for the mount point's files. This value is normally used in the site file.
extlink	Specifies the use of the external link command to create, process, and delete a symbolic link to an MVS data set. Is used with the **ln -s** command to create a symbolic link to an MVS data set. Is used with the **ls -l** command to display the attributes and contents of the symbolic link. Is used with the **rm** command to delete the symbolic link. This **extlink** attribute only applies to HFS file objects.
fastfilesize	For Direct Access data sets, PDSs, and non-system managed data sets, this specifies to get the file size from SPF statistics, if they exist. For more information, see "Using fastfilesize to Avoid Read-for-Size" on page 116. This attribute also applies to PDSEs, but does not apply to HFS files.
nofastfilesize	For Direct Access data sets, PDSs, and non-system managed data sets, this specifies to read the entire file or member to get the file size. Using this attribute might cause a noticeable delay when first accessing very large data sets. For more information, see "Using fastfilesize to Avoid Read-for-Size" on page 116. This attribute also currently applies to PDSEs.
fileextmap	Turns on file extension mapping. This option can be specified at the file command level for the client platforms that support passing of attributes. The default is nofileextmap.
nofileextmap	Turns file extension mapping off.
mapleaddot	Turns on mapping of a single leading "." from a client file name to a legal leading "$" on MVS. This option would normally be enabled for access by AIX and UNIX clients. This attribute does not apply to HFS data sets.
nomapleaddot	Turns off mapping of a single leading "." from a client to a leading "$" on MVS.
maplower	Turns on mapping of lower case file names to upper case when accessing files on MVS, and back when sending to the network. This option would normally be enabled for access by AIX or UNIX clients. This option only affects file names (high-level qualifiers and user catalog aliases). This attribute does not apply to HFS data sets.
nomaplower	Turns off mapping of lower case file names to upper case and back when using files on MVS. Exports data set entries are not translated to upper case when this option is specified in the default attributes. All **mount** requests are case sensitive.

Table 16. Processing Attributes (continued)

Processing Attribute	Description
readtimeout(*n*)	The amount of time in seconds before a data set is released after a read operation. *n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). The default is **90**. The server closes the file when the file times out. This attribute does not apply to HFS.
noreadtimeout	The data set is not deallocated after a read operation. For more information, see "Timeout Attributes" on page 96, following this table.
	The OS/390 NFS server uses DFSMShsm to recall or delete migrated files. The action that the server takes against the migrated files depends on which of the **retrieve** or **noretrieve** attributes is active. These attributes do not apply to HFS data sets.
retrieve	When the **retrieve** attribute is active, the server will recall the migrated file if necessary, upon an NFS_LOOKUP request for the file, depending on the files status. The server may be able to obtain the migrated files attributes without recall (see "Retrieve Attributes" on page 96 for additional information); if not the recall operation is started by the server. The server waits for the recall operation to complete if the file resides on DASD; if the file does not reside on DASD, the server does not wait for the recall operation to complete and returns a "device not available" message. You can try accessing the file later when the recall has completed.
retrieve(*wait*)	When the **retrieve(wait)** attribute is active, the server waits for the recall to finish.
retrieve(*nowait*)	When the **retrieve(nowait)** attribute is active, the server does not wait for the recall to finish, and immediately returns a "device not available" message. You can try accessing the file later when the recall has completed.
noretrieve	When the **noretrieve** attribute is active, the server does not recall the file, and can return "device not available" upon an NFS_LOOKUP, an NFS_READ, or an NFS_CREATE request for a file. For more information, see "Retrieve Attributes" on page 96, following this table.
setownerroot	Sets the user ID in a file's attributes to `root` for a superuser. This attribute can only be used with mount command and does not apply to HFS.
setownernobody	Sets the user ID in a file's attributes to `nobody`, for a superuser.
sidefile(*dsname*)	Specifies the name of the data set that contains the rules for file extension mapping purposes. If a side file name is specified in the attributes data set, then it is the default side file for the NFS server. A user can also specify an additional side file name during a mount operation to be used along with the default. The mapping rules will first be searched in the side file specified during the **mount** command and then the default side file is searched. To allow file extension mapping, a side file name must be specified either as a default or in the mount command. *dsname* is a fully-qualified MVS data set name without quotation marks. *sidefile* is only specified at the mount level. See GFSAPMAP in NFSSAMP for sample side file and syntax rules. This attribute does not apply to HFS.

Table 16. Processing Attributes (continued)

Processing Attribute	Description
sync	Specifies that data transmitted with the **write** request should be committed to nonvolatile media (for example, DASD) by the server when received.
async	The user can alternatively specify the **async** processing attribute to get improved performance. The **sync|async** attribute only applies to HFS file objects and the NFS version 2 protocol.
writetimeout(*n*)	Specifies the amount of time, in seconds, before a data set is released after a write operation.*n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). The default is **30**. The server closes the file when the file times out. All cached buffers are forced to disk. Normally **writetimeout** values are kept short because WRITE operations result in exclusive locking. However, for slow client machines with long pauses between writes, you should increase the **writetimeout** value. This attribute does not apply to HFS.
nowritetimeout	Specifies that the data set is not deallocated after a write operation. For more information, see “Timeout Attributes”, following this table.
xlat*(member_name)*	This attribute can be used to override the installation default translation table during file processing. *member_name* is the member name of the PDS or PDSE that contains the customized translation table. The system administrator defines this member name in the attribute data set, and PDS or PDSE in the startup procedure. This attribute is ignored if specified on the command line. See Customizing the Translation Table in the OS/390 NFS Customization and Operation Manual for more information. If a customized translation table is not specified in the attribute file or in the mount command, xlat() is displayed by the **showattr**client enabling command.

Timeout Attributes

The values of the following attributes depend on the settings of the associated site attributes:

- The attributes **attrtimeout**, **readtimeout**, and **writetimeout** must be within the ranges specified by the **maxtimeout** and **mintimeout** site attributes.
- The attributes **noattrtimeout**,**noreadtimeout** and **nowritetimeout** are valid only when **nomaxtimeout** is specified in the site attributes.

Note: For HFS the default for **readtimeout** and **writetimeout** is 60.

There are three processing attributes which control when files are timed out: **attrtimeout**, **readtimeout**, and **writetimeout**. The server determines that of these timeouts are in effect based on the last file operation. Thus when an existing file is appended, the file cannot be accessed before it times out in the time specified for **writetimeout** and is released by the server, because write operations result in exclusive locking. Similarly, if a file is read, it is not released before it times out in the time specified for **readtimeout** seconds.

Retrieve Attributes

The server deletes the migrated file upon an NFS_REMOVE request for a file, regardless of whether the **retrieve** or the **noretrieve** attribute is active. Typically, an

NFS_REMOVE request is preceded by an NFS_LOOKUP request. If the dataset was migrated with DFSMS/MVS 1.2 or below, retrieve attribute causes a recall because NFS_LOOKUP processing needs to open the dataset and read for size. If the dataset was migrated under DFSMS/MVS 1.3 and DFSMShsm 1.3, and is SMS managed, its attributes were saved on DASD; therefore it is not always necessary to recall the data set to read for size and the dataset may be deleted without recall. If the **noretrieve** attribute is active, the NFS_LOOKUP can return a “device not available” message. If the client code decides to ignore the error and go with the NFS_REMOVE, the migrated file is then deleted.

The UNIX command **ls mvshost** does not issue requests for individual files under the `mvshost` directory. Migrated files under the `mvshost` directory are displayed, but are not recalled. However, the UNIX command **ls -l mvshost** issues NFS_LOOKUP requests for individual files under the `mvshost` directory.

Site Attributes

The site attributes are used to control OS/390 NFS server resources. These attributes are described in Table 17. Some initial settings are shown, but the system administrator might have changed these settings, so use the **showattr** command to show the actual settings being used. The site attributes cannot be modified by client users.

Table 17. Site Attributes

Site Attribute	Description
bufhigh(*n*)	Specifies the maximum size (in bytes) of allocated buffers before buffer reclamation (see the **percentsteal** attribute in this table) is initiated. *n* is an integer from 1MB to 128MB (the default is **32MB**). If the combined total specified in the **bufhigh** and **logicalcache** attributes is greater than the available storage in the extended private area (implied by the REGION parameter in your procedure) at startup, the server shuts down immediately. A higher number means more caching and potentially better read performance.
cachewindow(*n*)	Specifies the window size used in logical I/O to buffer client block writes received out of order. *n* is a number from 1 to 256 (the default is **112**). This attribute does not apply to HFS data sets. The suggested value is some small multiple of the number of BIODs running on a client. The general rule in setting the n value of cachewindow(n) is n = ((num of BIOD + 1) * (client_max_IO_buffer_size/transfer_size)) where, • BIOD is the number of blocked I/O daemons set by the client workstation. This value is usually set to defaults at the installation of the operating system or by your system administrator. • client_max_IO_buffer_size is the amount of I/O data requested by the client (for example, client writes 8192 bytes of data to the remote file system). This value is determined by your application programs. • transfer_size is the actual size of data being sent across the network (for example, the 8192 bytes of data can be broken down to 16 smaller packets of 512 bytes (16x512=8192)). This value is determined dynamically by your client workstation.

Table 17. Site Attributes (continued)

Site Attribute	Description
checklist	When specified, the server will bypass SAF checking (even when SAF or SAFEXP is specified) for the list of files and directories underneath mount points which either match a mount point entry or is a child of a mount point entry in the CHKLIST DD data set. CHECKLIST is only valid if SAF checking is the security option for the particular data access. It is ignored even if specified. See GFSAPCHK in NFSSAMP library for sample CHKLIST data set.
nochecklist	When specified, the server will operate as before and ignore the information specified in the CHKLIST DD data set.
hfs(*prefix***)**	Specifies a new HFS file system *prefix* to be imbedded in the mount directory path name. The default value of the HFS file system *prefix* is **/hfs**. Mount requests received by the OS/390 NFS server beginning with the HFS file system *prefix* value are identified as mount requests for OS/390 UNIX. The HFS file system *prefix* value is not part of the path name. **Notes:** 1. The HFS file system must be mounted locally by OS/390 UNIX or the client mount fails. 2. The *prefix* value can only be 7 characters or less including the beginning ″/″
nohfs	Disables Network File System OS/390 UNIX operation.
leadswitch	Tells the server to return '/' as the first character in each export entry.
noleadswitch	Tells the server not to return '/' as the first character in each export entry. **Note:** The **leadswitch** attribute is ignored for HFS file objects.
logicalcache(*n***)**	Specifies the maximum size (in bytes) of allocated buffers in the logical I/O processing for all the cache windows combined. *n* is an integer from 1MB to 128MB (the default is **16MB**). The recommended value for this attribute is 50% of the **bufhigh** attribute. This attribute does not apply to HFS data sets.
logout(*n***)**	Specifies the time limit for inactivity in seconds for a given user on a client (the default is **1800**). When the limit is reached, the user is automatically logged out. The client user must enter the **mvslogin** command again to reestablish the client's MVS session. This value should normally be the same as the value defined for TSO logout at your site. *n* can range from 1 second to 20 megaseconds (approximately 243 days).
maxrdforszleft(*n***)**	Defines the number of physical block buffers left after determining a file's size. This operation is done for later server read requests to the same file. The buffers left are subject to trimming during a "buffer steal" operation. *n* is an integer from 1 to 1024 (the default is **32**).

Table 17. Site Attributes (continued)

Site Attribute	Description
maxtimeout(*n*)	Specifies the maximum timeout allowed. This attribute and the **mintimeout** attribute define the range of values that client users can specify for **attrtimeout**, **readtimeout**, and **writetimeout**. *n* is the number of seconds from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds).This attribute does not affect the **logout** attribute.
nomaxtimeout	Allows client users to specify **noattrtimeout**, **noreadtimeout**, and **nowritetimeout**.
mintimeout(*n*)	Specifies the minimum timeout. This attribute and **maxtimeout** define the range of values that can be specified for **attrtimeout**, **readtimeout**, and **writetimeout**. *n* is the number of seconds from 1 to 32,767 (the default is **1**).
nfstasks(*n,m*)	Specifies the number of server processes to initiate on startup. The valid value range for both *n* and *m* is 1 to 24 (8 is the default for each). The sum of *n* and *m* must be less than or equal to 25). The absolute and relative value of *n* and *m* should be tuned for the expected system usage. If conventional MVS data sets will be accessed, primarily, then *n* should be relatively high. If OS/390 UNIX files will be accessed, primarily, then *m* should be relatively high. The absolute value of these will influence the amount of system resources consumed (higher values will make more system resources available to process NFS request. The parameter *n* is the number of processes started to handle asynchronous operations (conventional MVS data set access) and short duration synchronous operations (SAF calls). The parameter *m* is the number of processes started to handle longer duration synchronous operations (OS/390 UNIX file access and migrated MVS data set recall).
pcnfsd	Tells the Network File System to start the PCNFSD server.
nopcnfsd	Tells the Network File System not to start the PCNFSD server.
percentsteal(*n*)	Specifies the percent of the buffers reclaimed for use when the **bufhigh(*n*)** limit has been reached. A higher value means a reclaim operation is performed less often, but the cached data is significantly trimmed on each reclaim. This can result in poor read performance because readahead buffers might be stolen. Lower values result in more frequent reclaim operations, but the cached data normal water mark is higher, meaning possibly better performance by reading out of cached data. *n* is an integer from 1 to 99 (the default is **20**).
public(legacy_path,hfs_path)	This attribute specifies the legacy path (MVS conventional data) and HFS path that is associated with the public file handle for WebNFS access. The first path, if specified, is the legacy path. The second path is the HFS path and must start with the HFS prefix specified in the HFS() keyword. If the first path is not there, a comma must precede the second path. If the PUBLIC keyword is specified, then one of the paths must be specified. The PUBLIC keyword must be specified after the HFS() keyword in the site attribute table. A LOOKUP request with the public file handle determines which of the two paths it is referring to by the pathname that it comes in with. An absolute pathname will tell the server which of the paths it is referring to by matching one of the paths specified. A LOOKUP request with a relative pathname will be taken to be an HFS request if HFS is active (`hfs_path` has been provided); otherwise, it is treated as a legacy request. The default is **no public path**

Table 17. Site Attributes (continued)

Site Attribute	Description
readaheadmax(*n***)**	This attribute defines the number of bytes to be read to fill internal buffers during read processing to enhance satisfying read requests directly from cache. This reduces the amount of synchronous physical I/O performed for Network File System read requests for sequential read file access. It also reduces context switching overhead on Network File System read requests by allowing more read requests to be satisfied directly from the main task. *n* is an integer from 1KB to 128KB (normally 2 to 4 times the common block size used for file access, which is recommended at 8KB for AIX file activity). The default is **16,384**. Specifying zero (0) will deactivate readahead.
restimeout(*n,m***)**	Specifies a retention period and a clock time for the removal of mount points and control blocks that have been inactive longer than the specified retention period. *n* is the resource retention period for mounts and associated resources. If they have been inactive for more than *n* hours, they are removed. The valid range for *n* is 0 to 720 hours (30 days). The default is **48** hours. If *n* is set to 0, the OS/390 NFS server does not remove any mount points or associated resources. *m* is the time of day to do the cleanup for mounts and associated resources that have been inactive more than *n* hours. The time of day is specified as a 24 hour local time value. The valid range for *m* is 0 to 23. The default is **0** (that is, midnight). Because cleanup work slows down the server, set *m* so that cleanup work occurs when the server is lightly loaded. If a mount handle is removed by the cleanup activity, the user must do the **umount**and **mount** operations to access the mount point again. The resource cleanup is also done when the server is shutting down. This attribute does not apply to HFS data sets.
security(mvs,hfs,public)	Where mvs, hfs, public are place holders for security options: NONE, SAF, SAFEXP, EXPORTS. The first parameter, **mvs**, specifies the security option for MVS conventional data. The second parameter, **hfs**, specifies the security option for HFS data. The third parameter, **public**, specifies the security option for data accessed with the public file handle. The **mvs** parameter is required and the **hfs** and **public** parameters are optional. When the optional parameters are not specified, they are assigned the same security option as the first parameter. The security options are as follows: **none** Neither SAF checking nor exports list checking. For HFS, checks UNIX permission bits; obtains the UID from the client RPC request. **saf** SAF checking. No exports checking. For HFS, checks UNIX permission bits; obtains the UID from the OS/390 UNIX segment via **mvslogin**. **exports** Exports list checking. For HFS, checks UNIX permission bits for HFS data; obtains the UID from the client RPC request. No SAF checking. **safexp** SAF checking and exports list checking. For HFS, checks UNIX permission bits; obtains the UID from the OS/390 UNIX segment via **mvslogin**. The default is security(SAFEXP,SAFEXP, SAFEXP)
sfmax(*n***)**	Where *n* specifies the maximum size (in kilobytes) of allocated storage for all of the side files. *n* is an integer from 0 to 2000. The default value is 0 and it signifies that no mapping is allowed on the NFS server. If sfmax=0, specifying **sidefile** keyword in the attributes data set will cause the server to shut down and the specifying the sidefile in any subsequent mount commands causes the mount to fail as mapping is not allowed on the NFS server. If the amount of storage specified cannot be obtained during server initialization then the server will shut down immediately.

Table 17. Site Attributes (continued)

Site Attribute	Description
smf(none|user|file|userfile)	Specifies the SMF records to be recorded: **none** No SMF data collection **user** Collection of user session SMF data **file** Collection of file usage SMF data **userfile** Collection of file usage and user session SMF data

Appendix A. Messages to the Client

This appendix lists messages from three sources:

- The client's operating system, in response to Network File System reply results (messages without message numbers). These messages are platform-dependent. The message texts shown here are taken from AIX RS/6000.
- The Network File System, sent to the client from the OS/390 NFS server (messages with GFSA prefixed message numbers).
- The Network File System, sent to the client from the OS/390 NFS client (messages with GFSC prefixed message numbers).

The message format is "GFSAnnnt text", where:

GFSA	Component identifier for the OS/390 NFS server.
GFSC	Component identifier for the OS/390 NFS client.
nnn	Unique message number.
t	Message type:

A	Action; the user must perform a specific action.
E	Eventual action; the user must perform an action when time is available.
I	Informational; no user action is required.

text	Message text.

Messages from the Client Platform (AIX)

This sections contains messages from the client's operating system.

Cross device link

Explanation:

1. An attempt has been made to rename a member of a PDS or PDSE, but the target file is not a member of the same PDS or PDSE.
2. An attempt has been made to rename a non-PDS or PDSE file, but the target file is a member of a PDS or PDSE.

User Response: Try a copy and remove instead of a rename.

Directory Not Empty

Explanation: An attempt has been made to remove a PDS or PDSE that has members.

User Response: Delete members first before trying to remove a PDS or PDSE.

File exists

Explanation: An attempt has been made to rename a PDS or PDSE, but the target file already exists.

User Response: Delete the target file before renaming a PDS or PDSE. This is not required for a regular file or for a PDS or PDSE member.

File Name Too Long

Explanation: The name is not a valid MVS file or member name.

User Response: File names must follow the MVS naming conventions. See *DFSMS/MVS Using Data Sets* for MVS file naming conventions.

Invalid

Explanation:

1. **The parameters specified were incorrect**
2. **The creation of a VSAM data set failed**

User Response: Re-specify the correct parameters or contact the system programmer to determine the VSAM data set failure on the OS/390 NFS server.

I/O Error (with possible system programmer response)

Explanation:

1. Unexpected error from Catalog Management.
2. Dynamic Allocation failed during action other than read or write.
3. A file could not be opened during an action other than a read or write.
4. An error occurred while reading a partitioned data set (PDS) or PDSE directory.
5. No space is available in TIOT.
6. TIOT resource is unavailable.
7. Unable to release enough resources.
8. Insufficient units are available.
9. The server could not get enough memory to perform this function.

User Response: None

System Programmer Response: For (9), stop the server and change the region field in the job control language (JCL) before restarting, or modify parameters in the attributes file. For the other possible explanations, perform the appropriate action.

I/O Error (with possible user response)

Explanation:

1. An attempt has been made to nest PDSs or PDSEs.
2. Maximum number of file allocations is exceeded.
3. The file is being written in text mode, the new write request offset is determined to fall within the end-of-line (EOL) sequence (lf, cr, crlf, lfcr) of a previous line, and the new data does not contain the correct EOL characters.
4. The file is being written in text mode, with a non-zero EOL (lf, cr, lfcr, crlf). The number of bytes of data in the line written is larger than the maximum record size of the file.
5. The file is being written in text mode, with fixed records, with a non-zero EOL, **blankstrip** not set (no padding blanks on write), and the number of bytes of data in the line written is less than or greater than the record size of the file.
6. The file is being written in text mode, with fixed records, **blankstrip** set, and the line of data written contains trailing blanks as part of the data.
7. When a workstation file containing a 0 length line is written to MVS as recfm(u) in text mode, a write error occurs.
8. An MVS Access Method Services alias name was specified in a remove (**rm** or **rmdir**) or rename (**mv**) request.
9. An “s” was specified in the recfm attribute for a PDS or PDSE.
10. If you try to append data to a member of a PDS or PDSE, an I/O error occurs.
11. An incorrect attribute was specified in the command.

User Response: Perform the appropriate action.

Is a directory

Explanation: A non-directory operation has been tried on a PDS or PDSE.

User Response: Use directory operations on the file.

Network File System server *name* not responding still trying

Explanation: Long delays between operations.

User Response:

- The server might need extra time to service client requests. Wait until the message “Network File System server *name* ok” appears.
- Determine if the network traffic is heavy and overloaded. Try to isolate the path where the client workstation communicates with the server machine.
- Determine if the client workstation transmits the same requests over and over. Commands such as ’nfsstat -c’ on AIX or UNIX platforms show the number of client retransmissions (’retrans’) as well as the number of ’badcalls’ and ’badxid’. When the number of ’badcalls’ and ’badxid’ are high, the client machine usually has bad retransmissions. High retransmissions might be caused by an overloaded network or slow server.
- If the network is overloaded, contact your network administrator.
- If the server is slow, determine if your client workstation tends to transmit requests out of sequence or incompletely, and you are performing I/O to a file in text mode. If the request is incomplete, probably the remainder of the request does not get sent (by the client) until a much later time. In this case, increase the server’s cachewindow attribute. The ’cachewindow’ buffers store out of sequence and incomplete requests from clients. The general rule in setting the n value of cachewindow(n) is n = ((num of BIOD + 1) * (client_max_IO_buffer_size/transfer_size)) where,

  - BIOD is the number of blocked I/O daemons set by the client workstation. This value is usually set to defaults at the installation of the operating system or by your system administrator.
  - client_max_IO_buffer_size is the amount of I/O data requested by the client (for example, client writes 8192 bytes of data to the remote file system). This value is determined by your application programs.
  - transfer_size is the actual size of data being sent across the network (for example, the 8192 bytes of data may be broken down to 16 smaller packets of 512 bytes (16x512=8192)). This value is determined dynamically by your client workstation.
- If your client workstation tends to send duplicate transmissions for the same request too often, thus increasing the workload on the server, you might want to delay the client's retransmission rate and request's timeout. On the mount command, you can specify: `mount -o retrans=3,timeout=30 IO_buffer_size`, where:

  retrans is the number of retransmission allowed before a timeout. Default is vendor-specific ranging from 3-5.

  timeout
  is the timeout value in tenths of a second. timeout=30 means 3 seconds. Default is vendor specific ranging from 5-11.

  If the reply is not received by the client within the 'timeout' period, a 'minor timeout' has occurred for this request. The timeout period is doubled, and the request is sent again. The process is repeated until the retransmission count specified by the 'retrans' is reached; if no reply has been received, a major timeout has occurred.

No such device

Explanation:
1. The file resides on an off-line device.
2. The file has been migrated to another storage level. Whether it is being recalled depends on the **retrieve** attribute.

User Response: For (1), contact your MVS operator or MVS system programmer. For (2), try the request again later, if the **retrieve** attribute is enabled. If not, try the request again, with the **retrieve** attribute enabled.

No space left on device

Explanation:
1. The file has exceeded the space allocated to it.
2. The PDS or PDSE has exceeded the space allocated to it.
3. The PDS directory has exceeded the space allocated to it.

User Response:
1. For (1), save this file into a larger file and then rename it to the old name, if desired.
2. For (2), create a larger PDS or PDSE and store this member there. Then copy the members of the old PDS or PDSE to the new PDS or PDSE. Rename the new PDS or PDSE if desired.
3. For (3), Create a new PDS with a larger directory (use the dir attribute). Store this member in the new PDS. Then copy the members from the old PDS to the new PDS and rename if desired.

No such file or directory

Explanation: A locate failed for this file. The file is not cataloged, or the MVS system operator might have unmounted the file before you issued the **umount** command.

User Response: Check your spelling. If it is correct, contact your system administrator.

Not a directory

Explanation: A directory operation has been tried on a file that is not a PDS or PDSE.

User Response: Use non-directory operations on the file.

Not Owner

Explanation:
1. File has not timed out yet.
2. File is open. Another client (maybe even the same client) has the file open for writing.
3. The client tried to change the mode in a **nfsattr** Network File System procedure call.

User Response:
- For (1), follow the steps in waiting and retrying described under the “Permission denied” message.
- For (3), do not try to change the mode.

Permission denied

Explanation:
1. The file is not in use, but has not timed out yet (occurs most often when writing PDS or PDSE members without **writetimeout** seconds expiring between saving members).
2. The file is in use by an MVS user, or another client.
3. Dynamic allocation — Authorized function requested by an unauthorized user. The error codes from dynamic allocation are printed to the log data set.
4. Resource access control facility (RACF) is not active.

5. Dynamic allocation — Request denied by operator (dsname allocation). The error codes from dynamic allocation are printed to the log data set.
6. Dynamic allocation — Installation validation routine denied this request. The error codes from dynamic allocation are printed to the log data set.
7. You are not authorized for this request.
8. The MVS operator suspended mount request processing (only if seen following a mount attempt).
9. The file/prefix is not exported to your client (only if seen after a mount attempt).
10. IDCAMS failed during rename or remove. Usually this happens because the file is in use. The output from IDCAMS is printed to the log data set.
11. With OS/2, you might get a "SYS0055 Access denied" message if the **noretrieve** attribute is set and a **DIR** command is done against a mounted file system containing migrated files.

User Response: For (1), To determine if this is the problem: Check the timeout values (attrtimeout, readtimeout, writetimeout). Retry the request after the shortest timeout has expired. If the request still fails, retry the request after the next shortest timeout has expired. If it still fails, retry after the longest timeout has expired. If the request still fails, this is not the problem. For (2) and (10), try the request again later. For (3), (4), (5), (6), and (9), notify your MVS system programmer. For (7), enter the **mvslogin** command again and retry the request. If the request still fails, notify your MVS system programmer. For (8), notify your MVS operator or MVS system programmer. For (11), specify the **retrieve** attribute on the **mount** command, or the MVS system administrator can make that the default.

Read Only File System

Explanation: One of these Network File System procedures was tried on a read-only file system: link, write, rename, remove, mkdir, or create.

User Response: See the documentation on the exports data set to see how a file system is designated read-only (see *OS/390 Network File System Customization and Operation*). The exports data set needs to be changed or you are using it incorrectly.

Stale NFS File Handle

Explanation: A file handle is used by the client and server sides of the Network File System to specify a particular file or prefix. A stale file handle occurs when the name is no longer valid, possibly due to:

1. The file or prefix has been removed by the MVS operator.
2. The server has been stopped and brought back up. This affects files and members below mount points.

User Response: For (1), unmount and mount again. If your client maintains that the device is busy though it is not, you might have to restart your client. For (2), enter the **mvslogin** command again and retry the request.

Weak Authorization

Explanation: The authorization data in the remote procedure call (RPC) message was not valid. This is a client side error.

User Response: UNIX-style authorization is required.

Messages from the OS/390 NFS Server

This is a listing of the messages generated by the OS/390 NFS server. For each message, explanations and recommended actions are given where applicable. Data is substituted for any part of a message shown here in *italics*. Refer to "Messages from the OS/390 NFS Client" on page 109 for messages generated by the OS/390 NFS client.

GFSA950I Unknown flag '-*character*'

Explanation: The character *character* specified on the **mvslogin**, **mvslogout**, or **showattr** command is not a valid option. A usage message might follow this message.

User Response: See "Chapter 6. Commands and Examples for AIX and UNIX Clients" on page 39 for a description of the valid options used with the command.

GFSA951I *text*: can't find name for uid *d_digits*.

Explanation: There was an error reading information for UID *d-digits* from the **etc/passwd** file.

User Response: Correct the **etc/passwd** file and try the command again.

GFSA952I Retyped password does not match

Explanation: The password entered when message GFSA975I was displayed does not match the password entered when message GFSA974A was displayed.

User Response: Start the **mvslogin** command sequence again.

GFSA953I Password change required by host.

Explanation: The MVS password for the user ID passed to the host has expired. A new password is required. Message GFSA974I follows this message.

User Response: None.

GFSA954I Host *text1* **returned error** *d_digits***:** *text2*

Explanation: An error was detected during **mvslogin** processing. Host *text1* returned error code *d_digits* and message *text2* to the client.

User Response: The password or user ID might be incorrect. Start the **mvslogin** command sequence again and use the correct password or user ID.

GFSA955I *text* **logged in ok.**

Explanation: The MVS user ID *text* was logged in without any errors.

User Response: None.

GFSA956I usage: *text* **[-pn][-g group][-a account] hostname [mvs_username]**

Explanation: Usage information for the *text* command.

User Response: Enter the command using the correct syntax.

GFSA957I Host *text1* **returned error** *d_digits***:** *text2*

Explanation: An error was detected during **mvslogout** processing. Host *text1* returned error code *d_digits* and message *text2* to the client.

User Response: Notify your MVS system programmer.

GFSA958I uid *text* **logged out ok.**

Explanation: The MVS user ID *text* was logged out successfully.

User Response: None.

GFSA959I usage: *text* **hostname**

Explanation: Usage information for the *text* command.

User Response: Enter the command using the correct syntax.

GFSA960I *text1***: host** ″*text2*″ **unknown.**

Explanation: The host *text2*, specified on command *text1*, is not known to the network.

User Response: Correct the host name specified and try the command again.

GFSA961I *text1***:** *text2*

Explanation: Command *text1* received an error when trying to create a client transport handle using the **clntudp_create** TCP/IP remote procedure call. *text2* is the message produced by the **clnt_pcreateerror** TCP/IP procedure. This message is issued if:

- The host name is unknown
- The host is not operational
- The Network File System on the named host is not operational

User Response:

- Correct the host name specified and try the command again,
- Make sure the specified host is operational and try the command again, or
- Make sure the OS/390 NFS server on the named host is operational and try the command again.

GFSA964I *text***: Error: cannot determine server.**

Explanation: The *text* command found the mount path, but the server name was not returned by the local operating system service that keeps mount point information.

User Response: Correct the mount point table and try the command again.

GFSA965I *text1***: Error:** *text2* **mounted from server** *text3***, not** *text4***.**

Explanation: The wrong host name was specified for the *text1* command. *text2* is mounted from server *text3* instead of server *text4*.

User Response: Specify the command again with the correct host name.

GFSA966I *text***: Error: unknown return from usage routine.**

Explanation: The usage routine used for the *text* command returned an unknown error code.

User Response: Contact your programming support personnel.

GFSA967I Host Error: ***text*.**

Explanation: The host returned an error and message *text*. This might be due to:

- A porting failure occurred for the **showattr** command.
- The Network File System is not compatible with the **showattr** command on the client.

User Response: Contact your programming support personnel.

GFSA968I Error: Drive *text* **not mounted.**

Explanation: The drive *text* was not mounted. The **showattr** command cannot show the attributes for this drive.

User Response: Mount the drive and reissue the command.

GFSA969I Error: Can't open *text* **for read.**

Explanation: The file *text* could not be opened to read the mount path.

User Response: Correct the file and reissue the command.

GFSA970I Error: Directory *text* **not mounted.**

Explanation: The directory *text* was not mounted. The **showattr** command cannot show the attributes for this directory.

User Response: Mount the directory and reissue the command.

GFSA971I Error: filesystem *text* **is local.**

Explanation: The file system *text* is not a Network File System file system. The **showattr** command is for Network File System file systems only.

User Response: Reissue the command for a Network File System file system.

GFSA972I usage: *text1* **[-t] hostname [***text2***]**

Explanation: Usage information for the *text1* command. *text2* is the operating system dependent mount point format.

User Response: None.

GFSA973A Enter MVS password:

Explanation: The Network File System requires a password for the user.

User Response: If an MVS user ID was specified on the **mvslogin** command, enter the password for that MVS user ID. If no MVS_user ID was specified, the name from **etc/passwd** for the UID that issued the **mvslogin** command was passed to the Network File System. Enter the MVS password for this user.

GFSA974A Enter new MVS password:

Explanation: The MVS password for the user ID passed to Network File System has expired.

User Response: Enter a new MVS password.

GFSA975A Retype new MVS password:

Explanation: MVS requires the new password to be entered twice for verification.

User Response: Enter the new MVS password again.

GFSA976I *text1***:** *text2*

Explanation: Command *text1* received an error when trying to create a client transport handle using the **clnt_call** TCP/IP remote procedure call. *text2* is the message produced by the **clnt_perror** TCP/IP procedure.

User Response: Contact your system administrator.

GFSA977I *text***:**

Explanation: Command *text* received an error when trying to create a client transport handle using the **clnt_call** TCP/IP remote procedure call. This message is followed by the message produced by the **clnt_perror** TCP/IP procedure.

User Response: Contact your system administrator.

GFSA978I *text* **logged in ok. Mismatch in uid/gid: OpenEdition uid is** *digit_1***, gid is** *digit_2***, client uid is** *digit_3***, gid is** *digit_4*

Explanation: The OpenEdition UID/GID does not match the client machine UID/GID. The authentication is successful and the message is for informational use only.

User Response: None

Messages from the OS/390 NFS Client

This is a listing of the messages generated by the OS/390 NFS client. For each message, explanations and recommended actions are given where applicable. Data is substituted for any part of a message shown here in *italics*. Refer to “Messages from the OS/390 NFS Server” on page 106 for messages generated by the OS/390 NFS server.

Table 18 can be used for initial translation of the reason code, *reasoncd*, information presented in messages.

Table 18. Parsing error (when reason code is ’6E01xxxx’)

Last 4 hex digits of *reasoncd*	**Reason**
F*xxx*	Unknown keyword
11*yy*	Host name
12*yy*	Path name
13*yy*	Keyword **acdirmax**
14*yy*	Keyword **acdirmin**
15*yy*	Keyword **acregmax**
16*yy*	Keyword **acregmin**
17*yy*	Keyword **cln_ccsid**
18*yy*	Keyword **srv_ccsid**
19*yy*	Keyword **hard**
1A*yy*	Keyword **soft**
1B*yy*	Keyword **retrans**
1C*yy*	Keyword **timeo**
1D*yy*	Keyword **wsize**
1E*yy*	Keyword **rsize**
1F*yy*	Keyword **retry**
2A*yy*	Keyword **vers**
21*yy*	Keyword **biod**
22*yy*	Keyword **bufhigh**
23*yy*	Keyword **delaywrite**
24*yy*	Keyword **readahead**
25*yy*	Keyword **attrcaching**
26*yy*	Keyword **datacaching**
27*yy*	Keyword **dynamicsizeadj**
28*yy*	Keyword **delim**
29*yy*	Keyword **xlat**

Notes:

xxx	Offset to the bad keyword
yy	Refer to Table 19 on page 110 for more details

Note: For more information, see appendix on return codes in *OS/390 Version 2 Release 6 Network File System Customization and Operation*.

Table 19 can be used for further translation of the reason code, *reasoncd*, information presented in messages.

Table 19. Parsing error (when reason code is from '6E0111yy' to '6E0129yy')

Last 2 hex digits of *reasoncd*	**Reason**
01	Null host name or null path name
02	Blank detected
03	Incorrect member name in the path name
04	Missing double quote
05	No member name found
06	Missing left parenthesis
07	Incorrect number
08	Number is larger than 2G
09	Incorrect multiplier, must be K, M, or G
0A	Missing right parenthesis
0B	The specified number is not within the allowable range
0C	Incorrect keyword parameter value
0D	Mutually exclusive keyword/option
0E	Keyword is not allowed in the mount option
0F	Keyword is not allowed in the installation parameter

GFSC840I usage: *text* **[-a] [-d] [-e] [host]**

Explanation: This is the usage for the **showmount** command. *text* is the command as entered by the user. The valid options are as follows:

-a Display all mounts in the format Hostname:Directory from **host** NFS server

-d Display only directory names of all mounts from **host** NFS server

-e Display the list of exported directories from **host** NFS server

GFSC841E Unknown host *text*

Explanation: The user entered incorrect host address information, *text*.

System Action: Command stops processing.

User Response: Correct syntax and re-issue the command.

GFSC842E Cannot resolve local host name

Explanation: Local host name is not found.

System Action: Command stops processing.

User Response: Contact your system administrator to check TCP/IP configuration.

GFSC843E Unknown flag '-*character***'**

Explanation: An incorrect option, '-*character*', is specified.

System Action: Command stops processing.

User Response: Correct syntax and re-issue the command.

GFSC845I usage: *text* **input output**

Explanation: This is the usage for the **os22mvs** and **mvs2os2** commands. *text* is the command as entered by the user. The valid parameters are as follows:

input
Absolute path name of the input file to be converted

output
Absolute path name of the output file

GFSC846E Cannot open input file, *text1***:***text2*

Explanation: Cannot open input file, *text1*. *text1* is the input path name as entered by the user. *text2* is the failure information returned when attempting to open the input file.

System Action: Command stops processing.

User Response: Check input file, *text1*.

GFSC847E Cannot open output file, *text1***:***text2*

Explanation: Cannot open output file, *text1*. *text1* is the output path name as entered by the user. *text2* is the failure information returned when attempting to open the output file.

System Action: Command stops processing.

User Response: Check output file, *text1*.

GFSC848E Cannot read input file, *text1***:** *text2*

Explanation: Cannot read input file, *text1*. *text1* is the input path name as entered by the user. *text2* is the failure information returned when attempting to read the input file.

System Action: Command stops processing.

User Response: Check input file, *text1*.

GFSC849E Cannot write output file, *text1***:** *text2*

Explanation: Cannot write output file, *text1*. *text1* is the output path name as entered by the user. *text2* is the failure information returned when attempting to write the output file.

System Action: Command stops processing.

User Response: Check output file, *text1*.

GFSC850E Input path name cannot be equal to output path name.

Explanation: Input path name cannot be equal to output path name.

Module: GFSCWO2M or GFSCWM2O

System Action: Command stops processing.

User Response: Correct syntax and re-issue the command.

GFSC854I usage: *text* **[-crnzm <mount point>]**

Explanation: This is the usage for the **nfsstat** command. *text* is the command as entered by the user. The valid parameters are as follows:

-c Display both NFS and RPC statistics about OS/390 NFS client

-n Display NFS statistics about OS/390 NFS client

-r Display RPC statistics about OS/390 NFS client

-z Initializes statistics to zero. This is for use by root user only and can be combined with any of the above options. Zero particular set of statistics after printing them.

-m Display the name of each NFS mounted file system

-m mount point
Display information of NFS mounted file system on the specified mount point

GFSC855E Must be a root user to issue '*character***' flag**

Explanation: The option, '*character*', can only be issued with the root authority.

System Action: Command stops processing.

User Response: Contact your system administrator to issue this command.

GFSC856E Network File System Client command, *text***, failed, return value -1 return code** *returncd* **reason code** *reasoncd*

Explanation: The command, *text*, failed.

System Action: Command stops processing.

User Response: Refer to*OS/390 UNIX System Services Messages and Codes* for a description of the return code, *returncd*. Refer to Table 18 on page 109 for further information on the reason code, *reasoncd*.

GFSC858E Directory *text* **not mounted.**

Explanation: The directory, *text*, was not mounted.

System Action: Command stops processing.

User Response: Issue '**nfsstat -m**' to view the list of active mount points. If the mount point does not exist, contact system administrator to mount the directory.

Appendix B. Related Protocol Specifications

IBM is committed to industry standards. The internet protocol suite is still evolving through Requests for Comments (RFC). New protocols are being designed and implemented by researchers, and are brought to the attention of the internet community in the form of RFCs. Some of these are so useful that they become a recommended protocol. That is, all future implementations for TCP/IP are recommended to implement this particular function or protocol. These become the *de facto* standards on which the TCP/IP protocol suite is built.

The Network File System is implemented as a set of RPC procedures that use External Data Representation (XDR) encoding to pass arguments between clients and servers. The Network File System is based on the following RFCs:

- **Internet Protocol**, RFC 791, J.B. Postel
- **NFS: Network File System Version 2 Protocol Specification**, RFC 1094, Sun Microsystems, Incorporated.
- **NFS: Network File System Version 3 Protocol Specification**, RFC 1813, Sun Microsystems, Incorporated.
- **Open Group Technical Standard Protocols for Interworking: XNFS, Version 3W**, Document Number: C702
- **RPC: Remote Procedure Call Protocol Specification Version 2**, RFC 1057, Sun Microsystems Incorporated.
- **RPC: Remote Procedure Call Protocol Specification Version 2**, RFC 1831, R. Srinivasan.
- **User Datagram Protocol**, RFC 768, J.B. Postel
- **WebNFS Client Specification**, RFC 2054, B. Callaghan.
- **WebNFS Server Specification**, RFC 2055, B. Callaghan.
- **XDR: External Data Representation Standard**, RFC 1014, Sun Microsystems, Incorporated
- **XDR: External Data Representation Standard**, RFC 1832, R. Srinivasan.

These documents can be obtained from:

```
Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA  22021
```

Many RFCs are available online. Hard copies of all RFCs are available from the Network Information Center (NIC), either individually or on a subscription basis. Online copies are available using an anonymous FTP to NIC at `nic.ddn.mil`. Once you have logged into the machine, you can download the files using the following commands:

```
cd rfc
get rfc-index.txt
get rfcnnnn.txt
(change type to image or binary, for example: type image)
get rfcnnnn.ps
```

The `.txt` suffix is used on the text format files. The `.ps` suffix is used on the PostScript format files. Substitute `nnnn` with the actual RFC number.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to `service@nic.ddn.mil` with a subject line of `RFC INDEX`.

RFCs are also accessible through the online document library at the following Network Information Center World Wide Web home page:

```
http://www.nic.ddn.mil
```

For more information, contact the Network Information Systems Center at internet address:

```
nic@nic.ddn.mil
```

Appendix C. Handling of the File Size Value

Many Network File System procedures (such as nfs_lookup and nfs_getattr) in the NFS protocol require the file size to be returned. This appendix explains some performance and accuracy considerations in obtaining the file size value.

The meaning of the file size value returned by the Network File System and how fast the file size is returned depends on:
- Whether you use **text** or **binary** processing mode
- The type of MVS data set being accessed
- If the data set is system-managed
- If you use **fastfilesize** processing

Storage of the File Size Value

How the file size value is stored affects how quickly files are accessed and depends on the type of MVS data set used.

System-Managed PS, VSAM, and PDSE Data Sets

For system-managed PS, VSAM KSDS, VSAM ESDS, VSAM RRDS, and PDSE data sets, the file size value is stored on DASD and is returned quickly.

Text and binary file size are saved on non-volatile storage (DASD) and maintained by the server for these data set types:
- Physical sequential (including striped)
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDSE members

These data sets must be SMS managed. When the Network File System accesses a data set for the first time, it performs a read-for-size to get the text or binary file size and stores this value on DASD. Subsequent file size requests from clients do not cause the server to read for size, thus improving performance. However, when the data set is modified outside the server by a non-Network File System application (for example, by the TSO editor), the stored file size could be incorrect. When the data set is accessed again by the server, read-for size must be done to determine the correct file size.

Migrated System-Managed Data Sets

DFSMS/MVS allows data set attribute accessibility for SMS managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3. Supported data set types are SMS managed PS, VSAM ESDS, VSAM KSDS, VSAM RRDS, PDS, and PDSE. Migrated PDS/PDSE members are not supported.

The OS/390 NFS server is able to obtain the attributes of a supported SMS managed migrated data set without recalling the data set. Attributes such as the record format and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a non-Network File System application (for example, by the TSO editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall must be done to determine the correct file size.

Non-System Managed, PDS, and DA Data Sets

The file size value for non-system managed data sets, PDS members, and DA data sets is cached in virtual storage until timeout but not written to DASD. Therefore, for these types of MVS data sets, the file size value is re-generated after the file is closed or after the server is restarted.

How the File Size Value Is Generated

When a file is first accessed (for example with **ls -l** or **dir**), usually the entire file is read to determine its size, except for **recfm(f)** or **recfm(fbs)** where the binary size can be computed without reading the file. If the file is a system-managed PS, VSAM, or PDSE member, both binary and text file sizes are stored on DASD, so that subsequent file size requests do not require the file to be read.

Binary file size can be quickly generated by using **recfm(f)** or **recfm(fbs)** to specify a fixed-length record format for the MVS data set. With this format type, the server pads the last logical record with binary zeros in **binary** mode processing, because MVS always expects complete logical records. If the application tolerates these zeros, using **recfm(f)** or **recfm(fbs)** allows the binary size to be computed quickly because the number of bytes can be computed from the number of blocks, which is stored by MVS.

If you need the exact file size and are using **binary** mode processing, map it to a variable-format, sequential data set on DASD so that the Network File System does not need to pad a partially filled last MVS logical record to a record boundary.

For reading small files or the beginning of files, the read-for-size might not add any processing time. As the file is being read for size, the beginning of the file is stored in the buffers set aside by the **maxrdforszleft** site attribute, until the buffers are full. When the application reads the beginning of the file, this read is fast because it reads directly from the buffer.

MVS stores the number of blocks (rather than the number of bytes) in an MVS file. For most files, therefore, without reading the entire file, the Network File System can only give an estimate of the number of bytes in the file, not the exact number of bytes in the file. Even when the server could get the exact byte count without reading the file, the file size could change depending on the file's processing attributes.

For example, selecting **text** mode processing introduces line terminators such as `'lf'`, `'crlf'`, or `'\n'` into the file, thus changing the perceived size of the file. As another example, suppose you select **text** mode processing with blank stripping enabled on a fixed-length record format file. That causes the server to remove trailing blanks from each record, again changing the perceived size of the file. In these examples, when you first request a file, the server must read the entire file to determine its exact size in bytes.

Using fastfilesize to Avoid Read-for-Size

If you can use an approximate file size for a PDS, PDSE, DA, or non-system managed data set, you can specify the **fastfilesize** attribute to improve performance. With this attribute, the server estimates the size without opening and reading the entire file.

PDS members
For PDS and PDSE members, the **fastfilesize** attribute gets the file size from SPF statistics if they exist; otherwise, a zero file size is returned.

DA data sets
For DA data sets, an approximate file size is calculated based on the device characteristics, the number of disk tracks in use, and the block size of the data set.

VSAM For non-system-managed VSAM data sets, the estimated size using **fastfilesize** is zero. Therefore, **cat** or **vi** won't show any data.

The **fastfilesize** attribute speeds up data set access by calculating approximate file sizes during data set access. Use this only when you are browsing through files (using the **ls** UNIX command or the **type** OS/2 command, for example) because some commands (such as **cp** or **copy**) might not work correctly if **fastfilesize** is set. When reading or modifying a data set, the **nofastfilesize** attribute should be used to ensure accurate results.

nofastfilesize

When you use the default, **nofastfilesize** attribute, the Network File System reads the entire file or member to get the file size. It stores the file size value in cache until timeout. If the server's default has been changed to **fastfilesize**, you can still use the **nofastfilesize** attribute to override it. For example:

```
$ ls -l "filename,nofastfilesize"
```

Using this attribute might cause a delay when first accessing very large data sets.

Note: When directly mounting on a fully qualified data set name and nofastfilesize is specified, the server must return the mount size as part of getting the attributes for the mount. This can slow down the completion of the mount command.

Appendix D. Handling of the Time Stamps

UNIX file attributes define the following time stamps:

atime	The last time the file was accessed (read)
mtime	The last time the file was modified (write)
ctime	The last time the file status was changed (chmod)

The Network File System handles time stamps differently for these types of data sets:
- System-managed PS data sets and system-managed VSAM data sets
- Direct Access data sets and non-system managed PS data sets
- Non-system managed VSAM data sets
- PDS and PDSE members

Time Stamps for System-Managed VSAM and PS Data Sets

For system-managed PS data sets and system-managed VSAM data sets, *atime* and *mtime* are fully maintained, and the *ctime* is set to the *mtime*.

Time Stamps for Non-System Managed PS and DA Data Sets

For non-system managed PS and DA data sets, consider the following:
- How time stamps are stored
- The requirements of your workstation programs
- The type of MVS data set used to store the file

Storing Time Stamps

For non-system managed PS and DA data sets, the Network File System temporarily stores the time stamps in virtual storage, but not on DASD. These cached attributes are purged when the file times out and closes or when the server is restarted. When the file is accessed again, the time stamps are re-generated.

Client Program Requirements

Some workstation-based utilities (such as **make**) rely on date and time stamps to determine whether to recompile. For example, **make** checks the update time of the object file with the source file and recompiles if the source has been updated. Before storing these types of files using the MVS server, examine them before moving them to ensure that these attributes are unimportant. In an environment which relies on such utilities, use system-managed PS data sets.

Generating Time Stamps

This is how the Network File System generates *atime* and *mtime* for non-system managed PS and DA data sets from the MVS dates:

```
atime
= mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

time_increment is either the server local time or 23:59 hours. If *reference_date* or *creation_date* is equal to the server local date, the server local time is added. Otherwise, a fixed value of 23 hours and 59 minutes is added.

If *reference_date* = 0 (that is, the file has not yet been referenced), *atime* and *mtime* are set equal to *ctime*.

Time Stamps for Non-System Managed VSAM Data Sets

The time stamps for these types of data sets are set to the current time.

Time Stamps for PDSs and PDSEs

An MVS PDS data set can act as a UNIX directory. Members of the PDS are files within the UNIX directory. When the directory is accessed by the client, the UNIX times are expected for each file.

Ordinarily, MVS does not maintain time stamps for members of a PDS. The UNIX time stamps here are generated from the MVS creation and reference dates of the PDS data set containing the members. This is how the time stamps for PDS members are generated:

```
atime
= mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

ISPF is an MVS product that does maintain some additional statistics for each member. They include the creation date and the last modification date and time.

If the ISPF time stamps are present for a PDS member, this is how the server generates the time stamps and initializes the UNIX times:

```
atime = mtime = modification_date + modification_time
ctime = ISPF_creation_date + time_increment
```

time_increment is either server local time or 23:59 hours as described for non-VSAM data sets.

The server also creates new ISPF statistics for PDS members created by the clients. The ISPF statistics are created even if existing members do not have statistics.

The time stamp information is saved in the PDS directory according to ISPF conventions. If STATS=ON was specified when the member was created, the server uses them to get more accurate attributes. Even if STATS=ON was not specified originally, the server writes back new time stamp information if the member is modified from the workstation.

Time stamp generation for a PDSE member is identical to that of a PDS with one exception. Accurate *mtime* of a PDSE member is returned to a client as the result of a file attribute request for that PDSE member.

Setting Time Stamps

Network File System clients can issue SETATTR requests to set the *atime* and *mtime* for a system-managed PS or VSAM data set. For PDSE members, setting *mtime* is allowed, but setting *atime* is not supported. PDSE member *mtime* is also maintained by PDSE access methods, so it is modified when a TSO user modifies the PDSE member.

Appendix E. Sample Program for LOCKD and STATD

```
/*********************************************************************/
/* The System V lockf function has the following values:            */
/*                                                                  */
/*    #include                                                      */
/*    int lockf(int fd, int function, long size);                   */
/*                                                                  */
/* where 'function' has one of the following values:                */
/*    F_ULOCK Unlock a previously lockd region                       */
/*    F_LOCK  Lock a region (blocking)                               */
/*    F_TLOCK Test and lock a region (nonblockig)                    */
/*    F_TEST  Test a region to see if it is lockd                    */
/*                                                                  */
/* The lockf function uses the current file offset ( which the      */
/* process can set using the lseek system call) and the size argument*/
/* to define the "record."  The record starts at the current offset  */
/* and extends forward for a positive size, or extends backwards for */
/* a negative size. if the size is zero, the record affected extends */
/* from the current offset through the largest file offset (the end  */
/* of file). Doing an lseek to the beginning of the file followed by */
/* a lockf with a size of zero locks the entire file.               */
/*                                                                  */
/* The lockf function provides both the ability to set a lock and to */
/* test if a lock is set.  When the function is F_LOCK and the region*/
/* is already locked by another process, the calling process is put  */
/* to sleep until the region is avaiable.  This is termed blocking.  */
/* The F_TLOCK operation, however, is termed a nonblocking call - if */
/* the region is not available, lockf returns immediately with a     */
/* value of -1 and errno set to either EAGAIN or EACCESS. Also, the  */
/* F_TEST operation allows a process to test if a lock is set,       */
/* without setting a lock.                                           */
/*                                                                  */
/* NOTE:  To do a nonblocking lock, we must use the F_TLOCK          */
/*        operation.  If we write                                    */
/*        ....                                                       */
/*        if (lockf(fd, F_TEST, size) == 0) {                        */
/*                rc = lockf(fd, F_LOCK, size);                      */
/*                ...                                                */
/*        }                                                          */
/*        There is a chance the some other process can issue lockf   */
/*        call between the F_TEST and the F_LOCK, which cause the    */
/*        F_LOCK to block. The F_TLOCK provides the ability to do    */
/*        the test and set a single operation, which is needed.      */
/*********************************************************************/
/* 4.3BSD Advisory Locking                                           */
/* The flock system call is provided to lock and unlock a "file"     */
/*                                                                  */
/*    #include                                                      */
/*    int flock(int fd, int operation);                             */
```

Figure 24. Sample Program for Record Locking and File Locking

```
/*                                                                    */
/* fd is a file descriptor of an open file, and operation is built   */
/* from the following constants:                                      */
/*                                                                    */
/*    LOCK_SH    Shared lock                                          */
/*    LOCK_EX    Exclusive lock                                       */
/*    LOCK_UN    Unlock                                               */
/*    LOCK_NB    Don't block when locking                             */
/*                                                                    */
/* More than one shared lock can be applied to a file at any time,   */
/* but a file cannot have both a single lock and an exclusive lock,  */
/* or multiple exclusive locks at any time. The permissible locking  */
/* operations are                                                     */
/*                                                                    */
/*    LOCK_SH              Shared lock(blocking)                      */
/*    LOCK_EX              Exclusive lock(blocking)                   */
/*    LOCK_SH | LOCK_NB    Shared lock(nonblocking)                   */
/*    LOCK_EX | LOCK_NB    Exclusive lock(nonblocking)                */
/*                                                                    */
/* If the lock was successful, zero is returned by flock, otherwise  */
/* -1 is returned. If the lock fails because the file is already     */
/* locked and a nonblocking lock was requested (LOCK_NB), the global */
/* errno is set to EWOULDBLOCK.                                       */
/*                                                                    */
/* NOTE:  must use "-lbsd" argument with the "cc" command in order   */
/*        to obtain the flock function for AIX                        */
/**********************************************************************/
#define F_ULOCK     0    /* Unlock locked sections          */
#define F_LOCK      1    /* Lock a section for exclusive use */
#define F_TLOCK     2    /* Test and lock a section for
                            * exclusive use                 */
#define F_TEST      3    /* Test section for locks by other
                            * processes                      */
main(int argc, char **argv)
{int                  fd, j;
 long               tloc;
 unsigned long    seconds = 10;
 pid_t                pid;
 int               status;
 if (argc < 2) {
   fprintf(stderr, "arg1 = file to lock.\narg2 = seconds to hold lock "
           "(default = 10 seconds).\n");
   return 101;}
 if (argc > 2) {  /* Parse seconds value. */
  if (1 != sscanf(argv[2], "%ld", &seconds)) {
    fprintf(stderr, "%s:  cannot understand arg2 value (%s).\n",
     argv[0], argv[2]);
    return 109;} }
```

```
time(&tloc);        /* Get current time. */
printf("Starting %s on file \"%s\" at %s",
argv[0], argv[1], asctime(localtime(&tloc)));
fd = open(argv[1], O_RDWR);
if (fd < 0) {
  perror("open failed");
  return 102;}
printf("%s:  calling flock [LOCK_SH].\n", argv[0]);
j = flock(fd, LOCK_SH);
if (j) {
  perror("flock failed");
  return 103;}
 printf("%s:  flock successful; sleeping for %d seconds...\n", \
        argv[0], seconds);
 j = sleep(seconds);
 if (!j) fprintf(stderr, "%s:  exit code %d from call to sleep for %d "
                "seconds.\n",argv[0], j, seconds);

 printf("%s:  calling flock [LOCK_UN].\n", argv[0]);
 j = flock(fd, LOCK_UN);
 if (j) {
   perror("flock failed");
   return 103;}
 else printf("%s:  flock successful.\n", argv[0]);

 printf("%s:  calling flock [LOCK_EX].\n", argv[0] );
 j = flock(fd, LOCK_EX);
 if (j) {
    perror("flock failed");
    return 103;}

 printf("%s:  flock successful; sleeping for %d seconds...\n", \
         argv[0], seconds);

  j = sleep(seconds);
  if (!j) fprintf(stderr, "%s:  exit code %d from call to sleep for %d "
                "seconds.\n",argv[0] , j, seconds);

  printf("%s:  calling flock  [LOCK_UN] .\n", argv[0] );
  j = flock(fd, LOCK_UN);
  if (j) {
    perror("flock failed");
    return 103;}
  else printf("%s:  flock successful.\n", argv[0]);

  if (-1 == lseek(fd, 0, SEEK_SET)) {
    perror("lseek to 0 error");
    goto end;}
  printf("%s:  calling lockf [lock at 0 for 3].\n", argv[0]);
```

```
if (lockf(fd, F_LOCK, 3)) {
    perror("lockf error");
    goto end;}
  if (-1 == lseek(fd, 10, SEEK_SET)) {
    perror("lseek to 10 error");
    goto end;}
  printf("%s:  calling lockf [fock at 10 for 3] .\n", argv[0]);
 if (lockf(fd, F_LOCK, 3)) {
    perror("lockf error");
    goto end;}
  pid = fork();
  if (pid == 0) {   /* New process; test locks. */
    if (lockf(fd, F_TEST, 3) == 0) {
      printf("%s: lock test from new process succeeded when it ought "
      "to have failed.\n", argv[0]);
      return 1;}
    else {
      perror("New process correctly finds lock not available");
      return 0;} }   /* End of child process. */
  else {        /* Parent process. */
    if (wait(&status) != pid) {
      perror("Wait failed");
      goto end;} }

    if (lockf(fd, F_ULOCK, 3)) fprintf(stderr, "%s: unlock at 10 for 3 "
        "failed (errno = %d).\n", argv[0] , errno);
    if (-1 == lseek(fd, 0, SEEK_SET)) {
    perror("lseek to 0 error");
    goto end;}
    if (lockf(fd, F_ULOCK, 3)) fprintf(stderr, "%s: unlock at 0 for 3 "
        "failed (errno = %d).\n", argv[0], errno);
  end:   time(&tloc);        /* Get current
time. */    printf("Ending %s at %s",   argv[0] ,
asctime(localtime(&tloc));     close(fd);
```

List of Abbreviations

These abbreviations are defined as they are used in the OS/390 library. If you do not find the abbreviation you are looking for, see the *IBM Dictionary of Computing* , New York: McGraw-Hill, 1994.

This list can include acronyms and abbreviations from:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1).

A

AIX. Advanced interactive executive.

APAR. Authorized program analysis report.

APF. Authorized program facility.

ASCII. American National Standard Code for Information Interchange.

B

BIOD. Block input/output daemon.

BSAM. Basic sequential access method.

C

CCSID. Coded character set identifier

CDRA. Character data representation architecture.

D

DA. Direct access.

DASD. Direct access storage device.

DBCS. Double byte character set.

DCB. Data control block.

DOS. Disk operating system.

DSCB. Data set control block.

E

EBCDIC. Extended binary coded decimal interchange code.

EOR. End of record.

ESDS. Entry-sequenced data set.

ETR. Electronic technical response.

F

FMID. Function modification identifier.

FTP. File transfer protocol.

FUB. File usage block.

G

GID. Group ID

GXB. Global exit block.

I

ICF. Integrated Catalog Facility.

IDCAMS. Integrated catalog access methods services.

IP. Internet protocol.

IPC. Inter-process communication.

ISO. International Organization for Standardization.

J

JCL. Job control language.

K

KSDS. Key-sequenced data set.

L

LFS. Locical file system.

M

MBCS. Multiple byte character set.

MVS. Multiple virtual system.

N

NFS. Network File System.

| **NLM.** Network Lock Manager

| **NSM.** Network Status Monitor

O

| **OS/390 UNIX.** OS/390 UNIX System Services

P

PDS. Partitioned data set.

PDSE. Partitioned data set extended.

PFS. Physical file system.

POSIX. Portable operating system interface for computer environments. An IEEE operating system standard, closely related to the UNIX system (software writing).

PS. Physical sequential.

PTF. Program temporary fix.

Q

QSAM. Queued sequential access method.

R

RACF. Resource Access Control Facility.

RBA. Relative byte address.

RPC. Remote procedure call.

RRDS. Relative record data set.

S

SAF. Security authorization facility.

SMS. Storage Management Subsystem.

SPF. System productivity facility.

SSF. Software support facility.

T

TCB. Task control block.

TCP/IP. Transmission control protocol/internet protocol.

TIOT. Task input/output table.

TSO. Time-sharing option.

TSO/E. Time-sharing option extended.

U

UCB. Unit control block.

UDP. User datagram protocol.

UXB. User exit block.

V

VSAM. Virtual storage access method.

VSCR. Virtual storage constraint relief.

X

XDR. External Data Representation.

Glossary

The following terms are defined as they are used in the OS/390 NFS Library. If you do not find the term you are looking for, see the IBM Software Glossary: http://www.networking.ibm.com/nsg/nsg

This glossary is an ever-evolving document that defines technical terms used in the documentation for many IBM software products.

A

Access. To obtain computing services.

access method. (1) A mainframe data management routine that moves data between storage and an I/O device in response to requests made by a program. (2) The part of the distributed data management architecture which accepts commands to access and process the records of a file.

access permission. A group of designations that determine who can access a particular AIX or UNIX file and how the user can access the file.

ACS routine. A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for an MVS file.

address. The unique identifier assigned to each device or workstation connected to a network.

address space. The complete range of addresses in memory available to a computer program.

Advanced Interactive Executive (AIX). IBM's licensed version of the UNIX operating system.

AIX. See *advanced interactive executive.*

alias. An alternative name for an MVS user catalog, a non-VSAM file, or a member of a partitioned data set (PDS) or PDSE.

alias entry. An entry that relates an alias to the real entryname of a user catalog or non-VSAM data set.

allocation. Generically, the entire process of obtaining a volume and unit of external storage, and setting aside space on that storage for a data set.

American Standard Code for Information Interchange (ASCII). The standard code used for information exchange among data processing systems, data communication systems, and associated equipment. It uses a coded character set consisting of 7-bit coded characters.

APAR. Authorized program analysis report.

APF. Authorized program facility.

application programming interface (API). A formally defined programming language interface between an IBM system control program or a licensed program and the user of a program.

ASCII. See *American National Standard Code for Information Interchange.*

atime. The time when the file was last accessed.

authentication. (1) In computer security, verification of the identity of a user or the user's eligibility to access an object. (2) In computer security, verification that a message has not been altered or corrupted. (3) In computer security, a process used to verify the user of an information system or protected resources. (4) A process that checks the integrity of an entity. (5) See also identity validation, message authentication code, password.

automatic class selection (ACS). A mechanism for assigning Storage Management Subsystem classes and storage groups to data sets.

B

BIOD. The caching daemon that caches directory lookups and file data when remote files are accessed from the host.

BSAM. Basic sequential access method.

block. A string of data elements recorded, processed, or transmitted as a unit. The elements can be characters, words, or physical records.

C

CDRA. Character data representation architecture.

CCSID. Coded character set identifier

client. (1) A user. (2) A consumer of resources or services. (3) A functional unit that receives shared services from a server. (4) A system that is dependent on a server to provide it with programs or access to programs. (5) On a network, the computer requesting services or data from another computer.

client-server relationship. Any process that provides resources to other processes on a network is a *server.* Any process that employs these resources is a *client*. A machine can run client and server processes at the same time.

connection. An association established between functional units for conveying information.

current directory. The currently active directory; the directory that is searched when you enter a file name without indicating the directory that contains the file name. When you specify a file name without specifying a directory, the system assumes that the file is in the current directory.

D

daemon. A background process that is initiated at system start that continuously preforms a function required by another process.

DA. Direct access.

DASD. Direct access storage device.

DASD volume. A DASD space identified by a common label and accessed by a set of related addresses. See also *volume*, *primary storage*, *migration level 1*, *migration level 2*.

data set. In DFSMS/MVS, the major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. In OS/390 non-UNIX environments, the terms *data set* and *file* are generally equivalent and sometimes are used interchangeably. See also *file.* In OS/390 UNIX environments, the terms *data set* and *file* have quite distinct meanings.

data set organization. The way in which data is arranged within an MVS file (data set) on a mainframe.

DBCS. Double byte character set.

DCB. Data control block.

DES authentication. Requires a client to send credentials (its name, conversation key, window key, and a time stamp) to the server. The server then returns a verifier to the client. Data Encryption Standard (DES) credentials are sometimes called “secure” credentials because they are based on a sender’s ability to encrypt data using a common time reference. DES-style credentials require a randomly generated key used to encrypt a common reference time which is then used in turn to create a *conversation key.* DES-style credentials are less vulnerable to penetration than UNIX-style credentials because of the generated conversation key and the data that is encrypted with the conversation key. Other forms of attack (for example, replaying message traffic) are resisted because of the timestamp in the transmission. The Network File System does not support DES-style credentials.

DFSMSdfp. A DFSMS/MVS functional component or base element of OS/390, that provides functions for storage management, data management, program management, device management, and distributed data access.

direct access file. A type of MVS file for storing data on a random access device that is accessed using a record address.

directory. A file that maps the names of other directories and files to their locations.

DOS. Disk operating system.

DSCB. Data set control block.

E

EBCDIC. See *Extended binary coded decimal interchange code.*

end user. A person in a data processing installation who requires the services provided by the computer system.

entry-sequenced data set (ESDS). A VSAM file whose records are loaded without respect to their contents, and whose relative byte addresses (RBAs) cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the file.

EOR. End of record.

ESDS. See *Entry-sequenced data set.*

ETR. Electronic technical response.

exports data set. An MVS file on the server containing entries for directories that can be exported to Network File System clients. It is used by the server to determine which MVS files and prefixes can be mounted by a client, and to write-protect MVS files on the server.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

External Data Representation (XDR). A standard developed by Sun Microsystems, Incorporated for representing data in machine-independent format. XDR is a vendor independent way of representing the data. By using the XDR standard data representation convention, systems do not have to understand and translate every data format that exists on the network; there is only the one convention. Data is translated into XDR format before it is sent over the network and, at the reception point, is translated into the data convention used there. This means that new computer architectures can be integrated into the network without requiring the updating of translation routines. The new architecture simply includes a routine that translates its data format into XDR format and the new member of the network is ready to go.

Using XDR, data can be accessed or exchanged among machines of various hardware and software architectures without any translation or interpretation problems. Word lengths, byte ordering, and floating point representations appear to be the same to all nodes in the network.

F

file. A collection of information treated as a unit. In non-OpenEdition MVS environments, the terms `data set`and `file`are generally equivalent and are sometimes used interchangeably. see also `data set`

file handle. A file handle is used by the client and server sides of the Network File System to specify a particular file or prefix.

file system. In the AIX operating system, the collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk.

File Transfer Protocol (FTP). A TCP/IP protocol used for transferring files to and from foreign hosts. FTP also provides the capability to access directories.

FMID. Function modification identifier.

foreign host. Any host on the network other than the local host.

free space. Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence or for lengthening records already there; also, whole control intervals reserved in a control area for the same purpose.

FTP. See *File transfer protocol.*

FUB. File usage block.

G

gateway. A functional unit that interconnects two computer networks with different network architecture. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures.

GXB. Global exit block.

GID. See **group number**

group. (1) With respect to partitioned data sets, a member and the member's aliases that exist in a PDS or PDSE, or in an unloaded PDSE. (2) A collection of users who can share access authorities for protected resources.

group number (GID). A unique number assigned to a group of related users. The group number can often be substituted in commands that take a group name as an argument.

H

handle. (1) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened. (2) In the AIX operating system, the data structure that is a temporary local identifier on an object. Allocating a handle creates it. Binding a handle makes it identify an object at a specific location. (3) In the OS/400 application programming interfaces, a variable that represents an object.

hierarchical file system (HFS) data set. A data set that contains a POSIX-compliant file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using OS/390 UNIX System Services. See also*file system.*

host. A computer connected to a network that provides an access method to that network. A host provides end-user services.

I

ICF. Integrated Catalog Facility.

IDCAMS. Integrated catalog access methods services.

internet. See *internetwork.*

Internet. A specific internetwork that includes ARPANET, MILNET, and NSFnet. These networks use the TCP/IP protocol suite.

Internet Protocol (IP). The TCP/IP layer between the higher-level host-to-host protocol and the local network protocols. IP uses local area network protocols to carry packets in the form of diagrams to the next gateway or destination host.

internetwork. A collection of packet-switched networks that are connected by gateways. They work as a single network.

interoperability. The ability of hardware and software from different vendors to communicate on a network.

inter-process communication (IPC). Ways for programs to communicate data to each other and to synchronize their activities. Semaphores, signals, and internal message queues are common methods of inter-process communication.

IP. See *Internet protocol.*

IPC. See *Inter-process communication.*

ISO. International Organization for Standardization.

J

JCL. Job control language.

K

key-sequenced-data-set (KSDS). A VSAM data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by key access, and new records are inserted in the data set in key sequence because of `free`space allocated in the data set. Relative byte addresses of records can change because of control interval and control area splits.

KSDS. See *key-sequenced-data-set.*

L

LFS. Locical file system.

local host. The computer where a user's terminal is directly connected.

local network. That portion of a network physically connected to the host without intermediate gateways.

M

management class. A collection of management attributes, defined by storage administrator, used to control the release of allocated but unused space; to control retention, migration, and backup of data sets; to control the retention and backup of aggregate groups, and to control the retention, backup, and class transition of objects.

master catalog. A catalog that contains extensive data set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and accumulate usage statistics for data sets.

MBCS. Multiple byte character set.

migration level 1. DFSMShsm-owned DASD volumes that contain data sets migrated from primary storage volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage* and *migration level 2.*

migration level 2. DFSMShsm-owned tape or DASD volumes that contain data sets migrated from primary storage volumes or from migration level 1 volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage* and *migration level 1.*

mkdir. An AIX, UNIX, OS/2 and DOS command used to make a new directory.

mount. (1) The process of accessing a directory from a disk attached to the machine making the mount request (4.2 mount), or to the remote disk on a network (Network File System mount). (2) An operation that associates a group of files on a server with a directory (mount point) on a client to provide transparent access to the files through that directory. The files must be in a hierarchical arrangement.

mount handle data set. A data set used to store the file handles of mount points.

mount point. A directory established in a workstation or a server local directory that is used during the transparent accessing of a remote file.

mtime. The time of last data modification.

MVS. Multiple virtual system.

MVS/ESA. An MVS operating system environment that supports ESA/390.

mvslogin. A client command to connect your workstation to MVS.

mvslogout (or mvslogut). A client command to break the connection between your workstation and MVS.

N

network. (1) An arrangement of nodes and connecting branches. (2) A configuration of data processing devices and software connected for information interchange.

NFS. Network File System.

NLM. Network Lock Manager

NSM. Network Status Monitor

null credentials. Null credentials are usually associated with diskless workstations where it is impossible to obtain identifying information because there is no local (that is, local to the workstation) repository of information.

O

object storage hierarchy. A hierarchy consisting of objects stored in DB2 table spaces on DASD, on optical or tape volumes that reside in a library, and on optical or tape volumes that reside on a shelf. see **storage hierarchy**

optical volume. Storage space on an optical disk, identified by a volume label. See also *volume.*

| **OpenEdition MVS.** See *OS/390 UNIX System Services*

OS/390 UNIX. See *OS/390 UNIX System Services*

OS/390 UNIX System Services (OS/390 UNIX). The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the OS/390 operating system that allow users to write and run application programs that conform to UNIX standards.

OS/390 UNIX System Services file system. In the OS/390 UNIX environment, the collection of files and file management structures on a physical or logical mass storage device, such as a diskette, minidisk, or a disk drive. See also *HFS data set*.

P

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

PDS. See *partitioned data set*

PDS directory. A set of records in a partitioned data set (PDS) used to relate member names to their locations on a DASD volume.

PDSE. See *partitioned data set extended*

permission code. A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

permission field. One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

PFS. Physical file system.

port. (1) An endpoint for a communication between devices, generally referring to a physical connection. (2) A 16-bit number identifying a particular TCP or UDP resource within a given TCP/IP node.

Portmapper. A server that converts RPC program numbers into port numbers acceptable to the protocol. This server must be running to make RPC calls.

POSIX. Portable operating system interface for computer environments. An IEEE operating system standard, closely related to the UNIX system (software writing).

primary storage. A DASD volume available to users for data allocation. The volumes in primary storage are called primary volumes. See also *storage hierarchy*. Contrast with *migration level 1* and *migration level 2*.

protocol. (1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) A specification for the format and relative timing of information exchanged between communicating parties.

PS. Physical sequential.

PTF. Program temporary fix.

Q

QSAM. Queued sequential access method.

R

RACF. See *Resource Access Control Facility.*

RBA. See *relative byte address*

relative byte address (RBA). The displacement of a data record or a control interval from the beginning of the data set where it belongs; independent of the manner how the data set is stored.

relative record data set (RRDS). A VSAM file whose records are loaded into fixed-length or variable-length slots. Each slot has a unique relative record number. Data is placed in a specific slot based on a user-supplied relative record number.

remote procedure call (RPC). RPC is an independent set of functions used for accessing remote nodes on a network. Using RPC network services, applications can be created in much the same way a programmer writes software for a single computer using local procedure calls. The RPC protocols extend the concept of local procedure calls across the network, which means that distributed applications can be developed for transparent execution across a network.

Resource Access Control Facility (RACF). An IBM licensed program that is included in OS/390 Security Server and is also available as a separate program for the MVS and VM environments. RACF provides access control by identifying and verifying the users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

rmdir. An AIX, UNIX, OS/2 and DOS command used to remove (delete) a directory.

root. The user name for the system user with the most authority.

RPC. See *remote procedure call.*

RRDS. Relative record data set.

S

SAF. Security authorization facility.

sequential file. A type of MVS file that has its records stored and retrieved according to their physical order within the file. It must be on a direct access volume.

server. (1) A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server. (2) On a network, the computer that contains the data or provides the facilities to be accessed by other computers in the network. (3) A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system.

sharing. A term used in a computing environment to refer to utilizing a file on a remote system. It is done by mounting the remote file system, then reading or writing files in that remote system.

shell prompt. The character string on an AIX or UNIX command line indicating the system can accept a command (typically the $ character).

showattr. A client command used to display the values of the site, processing, and data set creation attributes.

SMS. See *Storage Management Subsystem.*

SPF. System productivity facility.

SSF. Software support facility.

stale file handle. A file handle is stale when the file handle for a file or prefix is no longer valid.

stateless. A stateless server can function correctly without maintaining any protocol state information about any of its clients. The NFS protocol is intended to be as stateless as possible. This avoids complex crash recovery; a client just resends requests until a response is received. The stateless protocol is chosen to minimize the probability of data losses due to a server crash.

storage hierarchy. An arrangement of storage devices with different speeds and capacities. The levels of the storage hierarchy include main storage (memory, DASD cache), primary storage (DASD containing uncompressed data), migration level 1 (DASD containing data in a space-saving format), and migration level 2 (tape cartridges containing data in a space-saving format). See also *primary storage*, *migration level 1*, *migration level 2*, and *object storage hierarchy*.

Storage Management Subsystem (SMS). A DFSMS/MVS facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

superuser. The user who can operate without the restrictions designed to prevent data loss or damage to the system (User ID 0).

T

tape volume. Storage space on tape, identified by a volume label, which contains data sets or objects and available free space. A tape volume is the recording space on a pingle tape cartridge or reel. see also `volume`

task control block (TCB). An MVS control block that describes an asynchronous process that is called a task.

TCB. See *task control block.*

TCP/IP. See *transmission control protocol/internet protocol.*

TIOT. Task input/output table.

transmission control block (TCB). An internal control block within the TCP/IP address space.

Transmission Control Protocol (TCP). A stream communication protocol that includes error recovery and flow control.

Transmission Control Protocol/Internet Protocol (TCP/IP). The two fundamental protocols of the Internet protocol suite. The abbreviation TCP/IP is frequently used to refer to this protocol suite. TCP/IP provides for the reliable transfer of data, while IP transmits the data through the network in the form of datagrams. Users can send mail, transfer files across the network, or execute commands on other systems.

TSO. Time-sharing option.

TSO/E. Time-sharing option extended.

U

UCB. Unit control block.

UNIX authentication. Requires a client process to send credentials (the client’s machine-name, uid, gid, and group-access-list) to the server.

UDP. User datagram protocol.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It lessens the contention for the master catalog and facilitates volume portability.

User Datagram Protocol (UDP). A connectionless datagram protocol that requires minimal overhead, but does not guarantee delivery.

user number (UID). In the AIX operating system, a number that uniquely identifies a user to the system. It is the internal number associated with a user ID.

UXB. User exit block

V

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access storage devices. The records in a VSAM file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the file (entry sequence), or by relative entry number.

volume. The storage space on DASD, tape, or optical devices, which is identified by a volume label. See also *DASD volume*, *optical volume*, and *tape volume*.

VSAM. See *virtual storage access method*

VSCR. Virtual storage constraint relief.

X

XDR. See **External Data Representation**.

Index

Readers' Comments — We'd Like to Hear from You

OS/390
Network File System
User's Guide

Publication No. SC26-7254-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC26-7254-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Buisness Machines Corporation
RCF Processing Department
G26/M86 050
5600 Cottle Road
SAN JOSE, CA 95193-0001

Fold and Tape **Please do not staple** Fold and Tape

SC26-7254-00

Cut or Fold
Along Line

File Number: S370-40
Program Number: 5647—A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-7254-00