IBM

Hierarchical File System Usage Guide

- Learn how to manage HFS data
- Optimize HFS performance
- Set up sysplex sharing

Nigel Morton
Toru Yamazaki
Rama Ayyar
Dirk Gutschke
Barry Kadleck
Masato Miyamoto

Redbooks

ibm.com/redbooks

International Technical Support Organization

SG24-5482-01

Hierarchical File System Usage Guide

September 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 307.

Second Edition (September 2000)

This edition applies to Version 1, Release 5 of DFSMS/MVS, Program Number 5695-DF1 for use with Version 2 Release 7 of OS/390 or higher.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

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

Contents

Figures

Tables

Preface

This IBM Redbook provides technical information that helps you to implement and manage the hierarchical file system (HFS) available for OS/390 UNIX System Services (USS).

This redbook is intended for experienced S/390 storage administrators who are familiar with system managed storage and DFSMS/MVS storage management functions.

First, it introduces basic HFS concepts, including evolutionary functional enhancements available in DFSMS/MVS Version 1 Release 5 and OS/390 USS for storage administrators.

Then, it discusses all the information you need to use and manage your HFS environment efficiently and effectively, such as HFS externals, storage management techniques, sharing rules, and performance guidelines.

Finally, it covers implementation considerations for selected OS/390 USS applications, such as Lotus Domino and WebSphere.

This redbook particularly applies to DFSMS/MVS 1.5 (program number 5695-DF1) for use with OS/390 2.7 and later.

We hope this redbook helps you establish your own HFS environment.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose Center.

We would like to thank the authors of the first edition of this book, Toru Yamazaki and Masato Miyamoto.

Nigel Morton is a project leader at the International Technical Support Organization, San Jose Center. He has a degree in Computer Science from Cambridge University and joined IBM UK in 1977 and has 20 years experience with storage hardware and software. Before joining the ITSO in 1999, Nigel worked at the Product Introduction Solutions Consultancy, Hursley, UK as a program manager.

Nigel may be reached at mortonn@us.ibm.com.

Rama Ayyar is a Senior IT Specialist with IBM Australia in Sydney. Rama holds a master's degree in Computer Science from the Indian Institute of Technology, Kanpur, India. He has 17 years experience with MVS, storage management, RACF and configuration management software and 27 years experience in the computer industry.

Dirk Gutschke is an IT Specialist working in the IBM Global Services PSS Software Support Center in Mainz, Germany. He provides level-2 technical support to customers and IBM personnel in the Central Region in EMEA. He has worked at IBM for 10 years. His areas of expertise include DFSMS/MVS with particular focus on PDSE and HFS.

Barry Kadleck is a Senior IT Specialist based in IBM Hursley Park, United Kingdom. He has worked at IBM for 15 years, specializing in ADSM (now Tivoli Storage Manager) for the last five years. He currently supports Early Support Programs for Tivoli Storage Solutions Products for customers throughout EMEA.

Thanks to the following people for their invaluable contributions to this project:

Jeffrey Berger
IBM Storage Systems Division

Michael Bull
IBM Storage Systems Division

Douglas Johnston
IBM Storage Systems Division

John Thompson
IBM Storage Systems Division

Simon Williams
IBM Australia

Michael Scott
IBM Storage Systems Division

Neil Shah
IBM Global Services

Dave Levish
IBM Software Solutions Division

William Schoen
IBM S/390 Software Development

Cecilia Lewis
IBM Storage Systems Division

Hirofumi Morita
IBM Japan

Mike Ebbers
International Technical Support Organization, Poughkeepsie Center

Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Tricia Jiang
Tivoli Storage Solutions Product Line

Emma Jacobs
International Technical Support Organization, San Jose Center

Yvonne Lyon
International Technical Support Organization, San Jose Center

Particular thanks are due to Gabrielle Velez from the ITSO in Rochester who edited this book.

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in“IBM Redbooks review” on page 325 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction to the Hierarchical File System

This chapter provides an overview of the hierarchical file system (HFS) and lists the enhancements provided in DFSMS/MVS version 1 release 5 and in OS/390 version 2 release 9.

1.1 HFS overview

An HFS data set is an MVS data set that contains a POSIX-compliant hierarchical file system, which is a collection of files and directories organized in a hierarchical structure that can be accessed using the OS/390 UNIX System Services (USS).

The file systems within HFS data sets have a tree structure based on a root directory with various subdirectories with files contained within directories. The files within an HFS data set are identified by their path and file names.

HFS is used by UNIX System Services and its applications such as ftp, NFS, WebSphere Application Server for OS/390, Domino/390, and Component Broker.

HFS data sets were introduced in MVS/ESA with DFSMS/MVS Version 1 Release 2, together with support for OpenEdition MVS. The support for HFS data sets was rewritten for DFSMS/MVS Version 1 Release 5 to increase the performance of an HFS.

Figure 1 illustrates the relationship between DFSMS/MVS HFS services and OS/390 USS.

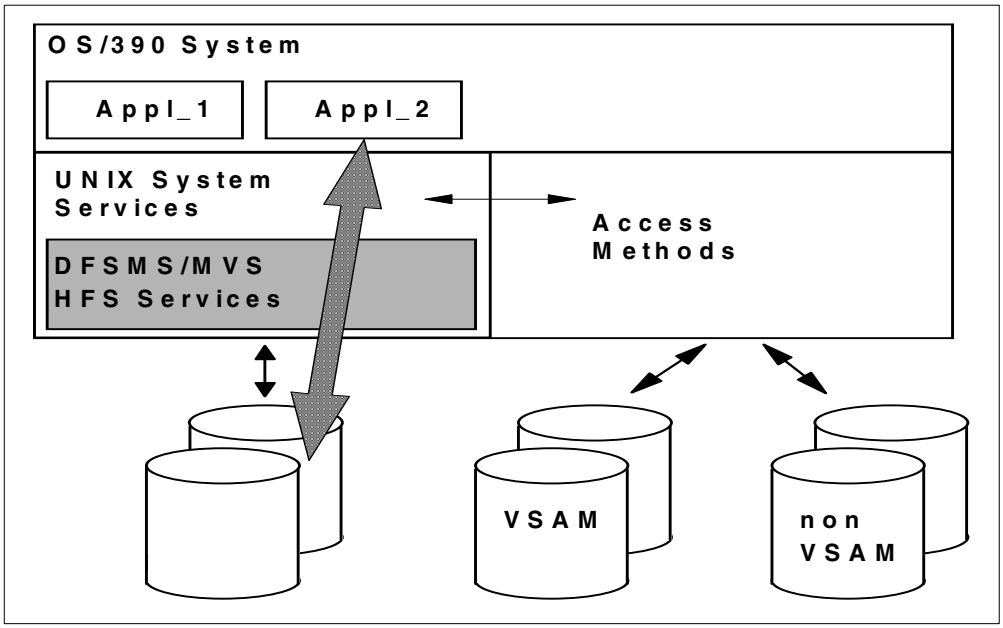


Figure 1. HFS overview

DFSMS/MVS Version 1 Release 5 is a base element of OS/390 Version 2 Release 7.

These are the basic characteristics of an HFS data set:

- DFSMS/MVS provides access to the files of data within an HFS data set.
- DFSMS/MVS provides storage management capabilities, such as backup/recovery and migration/recall for HFS data sets.
- HFS data sets can expand into multiple volumes (this support was provided in DFSMS/MVS 1.5). The single volume restriction for HFS data sets has been removed. HFS data sets can now span up to 59 volumes, with up to 255 total extents for all volumes, and up to 123 extents per volume.
- HFS data sets no longer must be SMS managed. Now, it is possible to allocate non-SMS managed HFS data sets.
- HFS data sets can be shared in read/write mode between systems in a Sysplex (the support was provided in OS/390 2.9).
- HFS data sets, not HFS files, have the following requirements and restrictions:
  - Non-SMS managed HFS data sets are supported but must be cataloged and single volume only.
  - They cannot be processed by standard open, end-of-volume, or access methods; POSIX system calls must be used instead. There is one exception: please refer to 3.4.2.6, “Processing HFS files with a sequential access method” on page 75.
  - They are supported by standard DADSM create, rename, and scratch functions.
  - They are supported by DFSMShsm for dump/restore and migrate/recall. DFSMSdss is used as the data mover.
  - They are not supported by IEBCOPY or the DFSMSdss COPY function.
  - Partial release for HFS data sets is not supported, since the DADSM PARTREL function no longer supports HFS data sets.

Note

Tivoli Storage Manager (formerly ADSM) can be used to back up, recover, archive or retrieve individual files or groups of files held in an HFS data set.

1.2 Structure of an HFS data set

All HFS data sets are stored on DASD volumes with fixed 4 KB blocks (4,096 bytes). These 4 KB physical blocks are also known as *pages*.

1.2.1 HFS track capacities on a 3380 and 3390 volume

Figure 2 shows the 3390 track capacity table (without keys). This table shows that 12 pages fit on one track for an HFS data set located on a 3390 volume.

Refer to *Using 3390 in an MVS Environment,*GC26-4574, for additional information regarding 3390 Direct Access Storage Devices (DASD).

Data Length Range		Percent Space	Maximum Track Capacity *		Maximum Cylinder Capacity *	
Min	Max	Used *	Records	Bytes	Records	Bytes
27 999	56 664	100.0	1	56 664	15	849 960
...	...	...	...	...	...	...
4 137	4 566	88.6	11	50 226	165	753 390
3 769	**4 136**	**87.6**	**12**	**49 632**	**180**	**744 480**
3 441	3 768	86.4	13	48 984	195	734 760
...	...	...	...	...	...	...

* Calculations are made using maximum size record in range.

Figure 2. 3390 track capacity table

According to the track capacity table for a 3390 volume in 3380 track compatibility mode, shown in Figure 3, one track of an HFS data set on a 3380 (or a 3390 in 3380 track compatibility mode) can hold 10 4KB pages.

Data Length Range		Percent Space	Maximum Track Capacity *		Maximum Cylinder Capacity *	
Min	Max	Used *	Records	Bytes	Records	Bytes
23 477	47 476	100.0	1	47 476	15	712 140
...	...	...	...	...	...	...
4 277	4 820	91.3	9	43 380	135	650 700
3 861	**4 276**	**90.0**	**10**	**42 760**	**150**	**641 400**
3 477	3 860	89.4	11	42 460	165	636 900
...	...	...	...	...	...	...

* Calculations are made using maximum size record in range.

Figure 3. 3380 track capacity table

This information may be helpful to calculate space and to interpret the different kinds of file size representation captured in 5.6, “Displaying the file system size” on page 143. Table 1 summarizes the maximum capacity for an HFS data set per tracks or cylinders for 3380 and 3390 volumes.

Table 1. HFS track and cylinder capacity

	3380		3390	
	Size in KB *	**Size in Pages ***	**Size in KB ***	**Size in Pages ***
1 Track	40	10	48	12
1 Cylinder	600	150	720	180
1 Volume	1,593,000 ***	398,250 ***	2,404,080 **	601,020 **

* includes data(files), index (AD) and directory
** 3390 Model3 (3,339 Cylinder per Volume)
*** 3380k (2,655 Cylinder per Volume)

1.2.2 Index structure of an HFS data set

HFS data sets, provided by DFSMS/MVS, contain the HFS file structure. This structure is a framework of directories and files called a file system.

Figure 4 provides an overview of the index structure of an HFS data set. The terms used in the following topics will be discussed again in 8.2, “RMF reporting enhancements” on page 192.

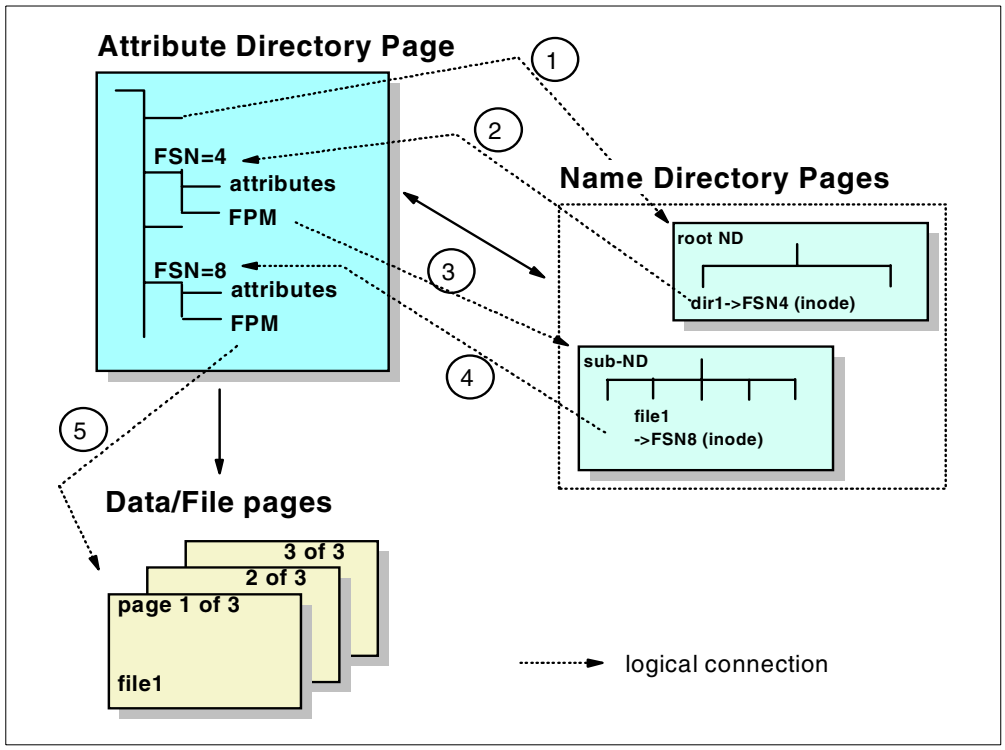


Figure 4. Index structure overview (with a single-level attribute directory)

This structure is maintained internally by different kinds of pages:

- **Attribute directory (AD) pages**

  The *attribute directory* is an internal hierarchical structure, implemented as a B-tree (a balanced tree where balancing minimizes the tree depth and so speeds up searching), which contains attribute information about individual file system objects such as HFS files, as well as attributes of the file system itself.

  The AD also provides the connection to the individual data pages that represent the files or directories inside an HFS. These connections are represented by a map called a *fragment parcel map* (FPM) — see callouts **(3)** and **(5)** within Figure 4.

- **Name directory (Root ND) and subname directory (subdirectory ND) pages**

  The *name directory* pages represent the external hierarchical directory of a file system. The ND also provides the connection between the individual file and directory names and the *file sequence number* (FSN).

  An HFS data set has one root ND page that is anchored at the AD — see callout **(1)**. Every directory below the root ND is represented by its own

subdirectory ND. These subdirectory NDs are also addressed by an FSN provided by their parent ND — see callout **(2)**. The associated AD entries (located by the FSN) provide the logical connection to the subdirectory ND pages — see callout **(3)**.

- **Data pages**

  The data pages contain the data of the individual files in a file system (HFS files). A file may contain data or an executable program. HFS files are tracked internally by a *file sequence number* (FSN) — see callout **(4)**. This number may be displayed by using the USS Command `ls -i` (list inode). The AD provides the connection to the data pages — see callout **(5)**.

The attribute, name, and name directory information is also referred to as *metadata*. Refer to Figure 5.

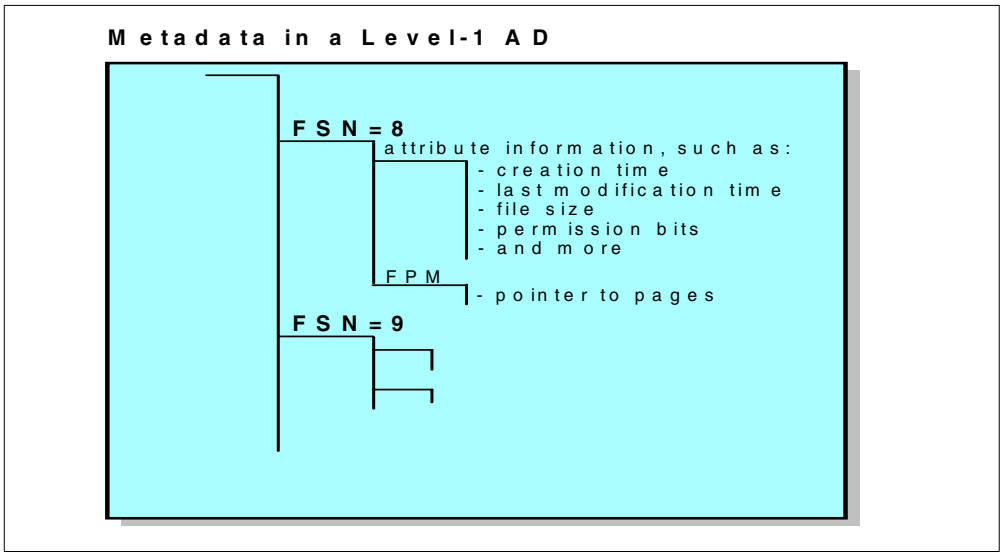


Figure 5. Metadata in a AD page

The AD, ND, and subdirectory ND are index structured (refer to Figure 6).

As files and directories are added to an HFS, the first level page AD or ND will eventually become full. When this happens, another level is added to the directory (this is known to as *new top processing*). The second level page is now the root page and it contains pointers to the first level page. In a similar fashion, if a second level index becomes full, another level is again added and the directory now has three levels. Only the first (lowest) level pages contain metadata. The second and higher level pages contain pointers to pages at lower levels. The first level pages are also known as the *sequence set pages*. The higher level pages are known as the *index set pages*. The highest level page is called the *root page of the index*.

For example, as shown in Figure 6 (second level AD structure), note that the root page contains no metadata, only pointers to the first (lowest) level pages.

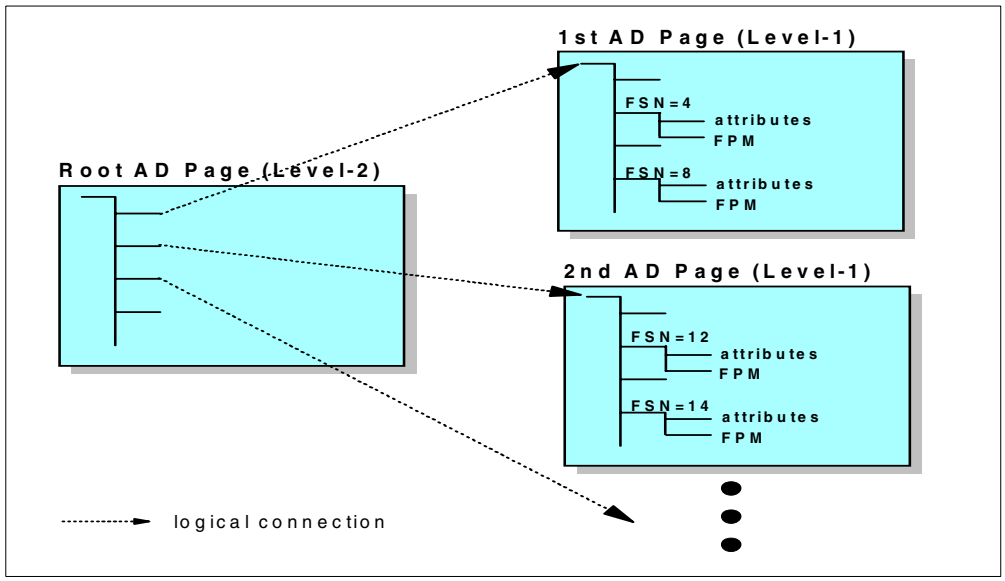


Figure 6. Second level AD index overview

The space in an HFS data set that is used for index information is obtained from the same storage (extents on a disk volume) that is available to the files themselves (data pages). This means that data pages and index pages are combined together in the same extents on a DASD volume.

Pages from deleted files (data pages) or index pages are available for reuse after the last connection to the file has been dropped.

The space used on a volume will only be formatted as the space is needed. This applies to the space for newly allocated HFS data sets, as well as the space for new extents. In DFSMS/MVS 1.4, the space was always formatted on a track boundary. In DFSMS/MVS 1.5, individual pages are fomatted rather than a complete track. The rest of the track is erased.The number of formatted pages will be stored inside the HFS data set. This value is also referred to as the *high formatted relative frame number* (HFRFN). The HFRFN is handled as a high watermark. This means that the HFRFN will not be decreased by deleting files or directories in a file system, because it represents the high formatted, not the high used, number of pages.

Important information

The HFRFN reflects the high watermark of formatted pages of an HFS data set and *not* the number of used pages inside an HFS data set.

It will not be decreased if files or directories are deleted.

You can use the new UNIX system services confighfs command to display the current value of the HFRFN. See 3.3.5, “confighfs shell command” on page 68 for more information about confighfs, and B.1, “HFS data set related information” on page 257, for sample output.

The backup and restore processing (done by DFSMSdss and DFSMShsm) is related to the HFRFN. Refer to Chapter 5, “Managing HFS data sets” on page

103 for more detailed information regarding HFS storage management provided by DFSMSdss and DFSMShsm.

1.2.2.1 Structure of a newly allocated HFS data set on DASD

An HFS data set always occupies at least five pages of storage. These five metadata pages are created at the same time as the data set.

Figure 7 illustrates the structure of a newly allocated HFS data set on a 3390 volume.

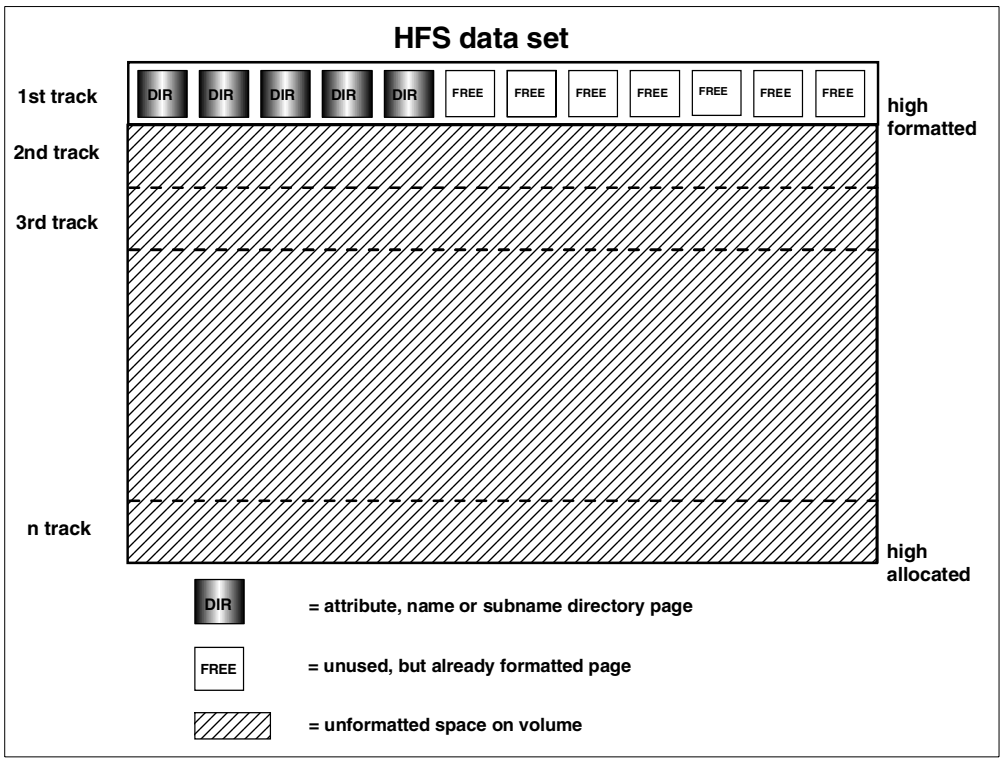


Figure 7. Structure of a new allocated HFS data set on a 3390 volume

1.2.2.2 Sample structure of a used HFS data set on DASD

Figure 8 shows a possible structure of an HFS data set stored on a 3390 volume. Directory pages, data pages and free pages are within the same extent. Unformatted space (*pages*) will be formatted automatically when they are needed.

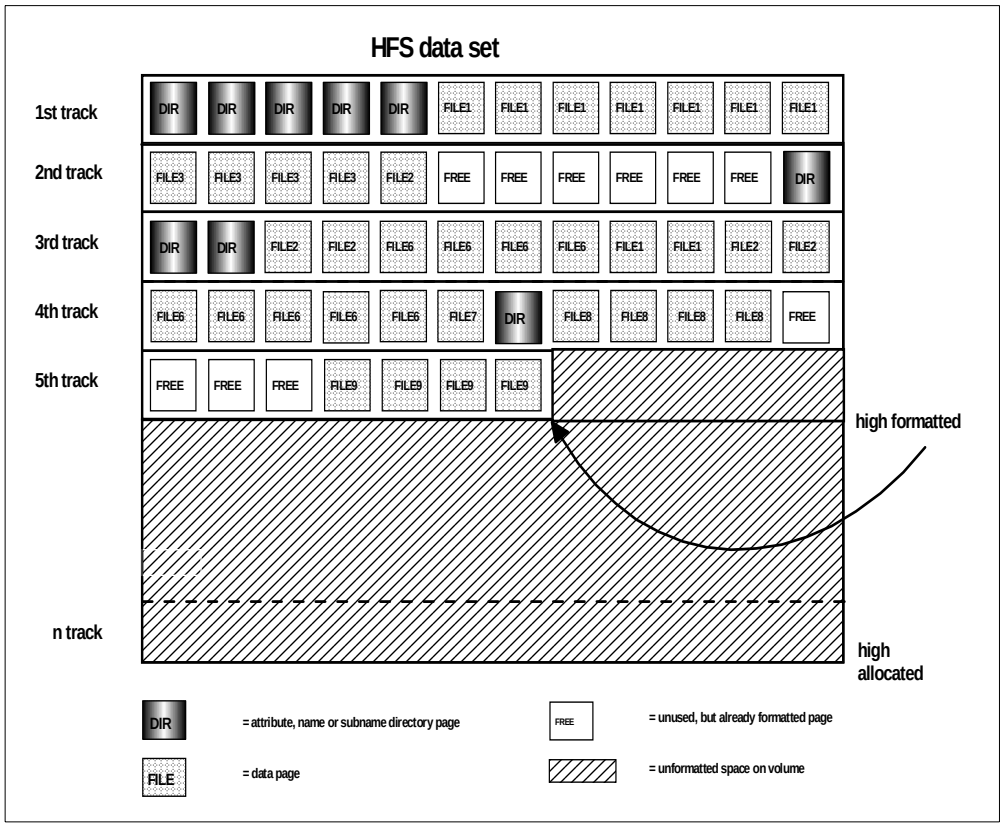


Figure 8. Sample structure of a used HFS data set on a 3390 volume

1.2.2.3 Sample of an empty HFS data set on DASD

The next figure shows an empty HFS data set. In this case, empty means that all files and all directories within that particular file system were deleted. However, the HFS data set still contains index (AD) pages. These pages are required to hold information about the HFS data set itself. Within the lifetime of an HFS data set, the AD pages will be split and merged together, but they will not be merged to the logical top of the HFS data set.

For example, as shown on Figure 9, index pages could reside on the last formatted track of an HFS data set.

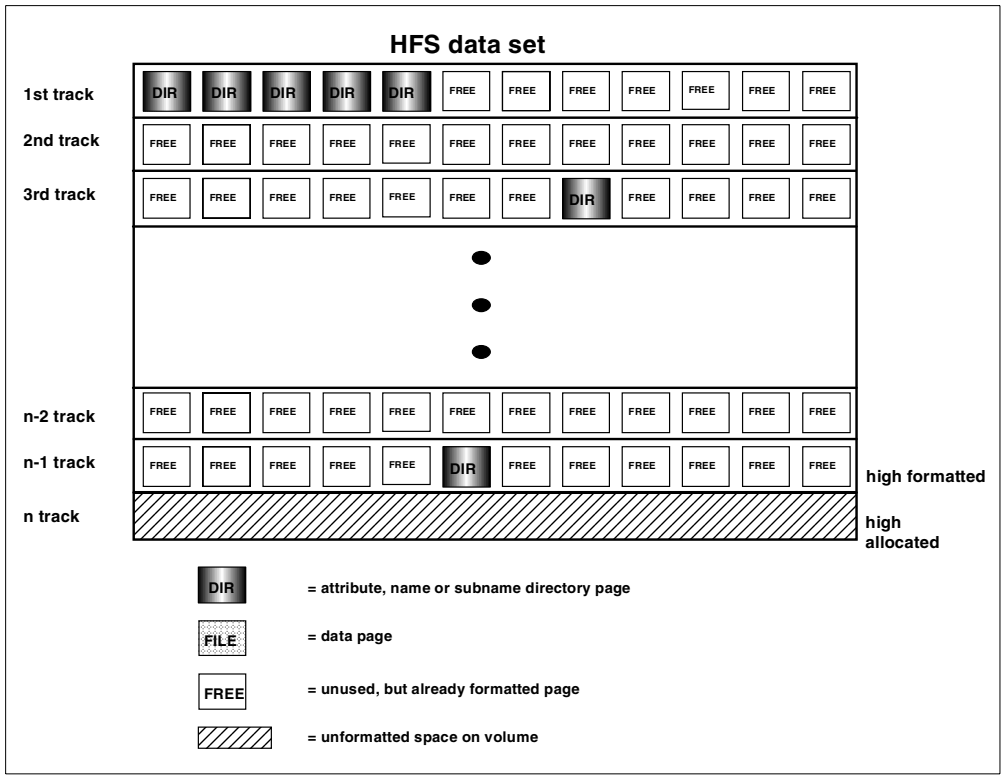


Figure 9. Sample structure of an empty HFS data set on a 3390 volume

Notes:

1. To reduce the space of an HFS data set below the high fomatted page, you can copy all files and directories individually into a new HFS data set by using UNIX commands. See the *OS/390 V2R9.0 UNIX System Services User's Guide,* SC28-1891, for additional information regarding copying files and directories.
2. Another possibility is that you can use the *copytree* utility provided by OS/390 USS. *Copytree* is a utility that can run under TSO or the shell, and is used to make a copy of a file hierarchy preserving all file attributes. It can be downloaded from an *OS/390 UNIX System Services* Internet page:

   `http://www.s390.ibm.com/oe/bpxa1ty2.html`

   Or, you can use:

   `http://www.s390.ibm.com/oe/`

   Select **Tools & Toys —> OS/390 Unix Tools**.

   You will also find additional information regarding copytree in 5.3.1, “Copytree utility” on page 127.
3. Or, as in this example, if the HFS data set is *empty*, you can allocate a new HFS data set.
4. Space reclamation was changed in DFSMS/MVS 1.5. In regard to both ADs and NDs, logic was added to merge pages together and return the now unused pages to the HFS for other files or directories to use. For NDs, there is also a function that will collapse the index down to a single level if it is empty. This process happens at every sync interval (the sync value can vary for each

individual HFS data set) for any HFS that requires an update to one of its files or directories. See 1.2.4, “Sync process” on page 11 for more details about the sync process.

1.2.3 Index updates through shadow writes

Every modification to the external file structure in an HFS will result in a change to the index (the internal hierarchical structure or AD), too.

For example, when you create new files or directories, or if you delete existing files or directories in an HFS, this will cause changes to the index as well as the data.

DFSMS/MVS HFS services use a shadowing process to update the index (AD) structure with new information. This means, if a *sequence set page* of the index structure must be changed, then a copy will be created instead of updating the current one. Afterwards, this copy page will be connected to its parent page within the index structure.

As the index (AD and ND) is maintained in a hierarchical tree structure, every change to a *sequence set page* in a B-tree results in a change to its parent pages (until the root page is reached), too. See Figure 10 for an example.

One benefit of the s*hadow write update technique* is: if the top (last) write is not performed, data remains consistent at the previous update level. This prevents an HFS data set from being damaged if, for example, a system crash occurs during the sync process. Updates may be lost, but the HFS will not be corrupted.

After the top is written to disk, all of the old pages that were replaced with new pages are freed for reuse.

An important point to note about this process is that, during the time that the index (AD) is being updated, the index temporarily requires the use of extra pages from the file system. This means that the file system must never be completely filled. The index usually holds some pages in reserve to make sure it has room to complete updates.

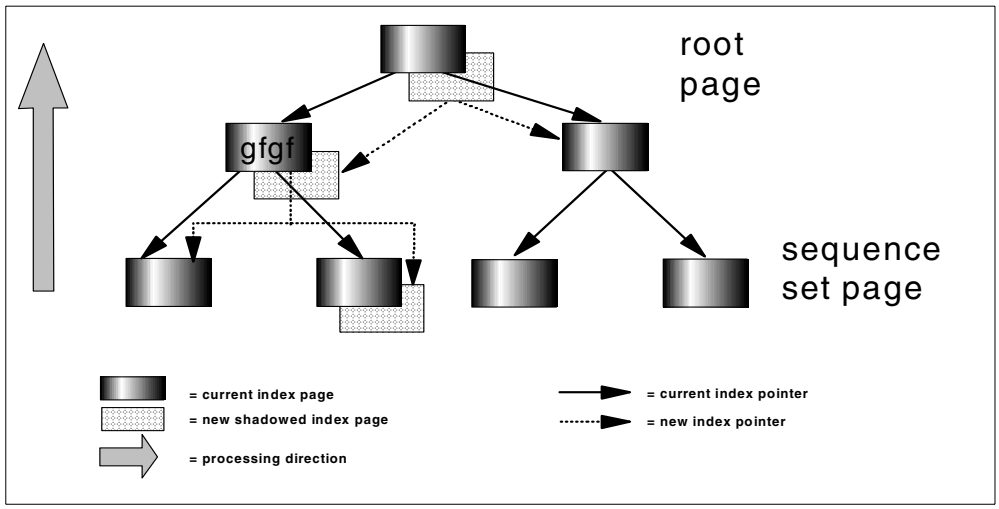


Figure 10. Shadow writing

Important Information

The *shadowing process* requires free pages in an HFS data set. Therefore, an HFS should never be completely filled.

1.2.4 Sync process

In DFSMS/MVS 1.4 and older releases, HFS buffer management was performed in the SYSBMAS *address space*, which is also primarily used by PDSE services. The index information was cached in a *data space* called SYSBMFDS. Any changes to the index were directly written to DASD under the user's task control block.

Beginning with DFSMS/MVS 1.5, HFS buffer management is performed within OMVS by a new function called the *sync daemon*. Figure 11 provides an overview of sync daemon processing.

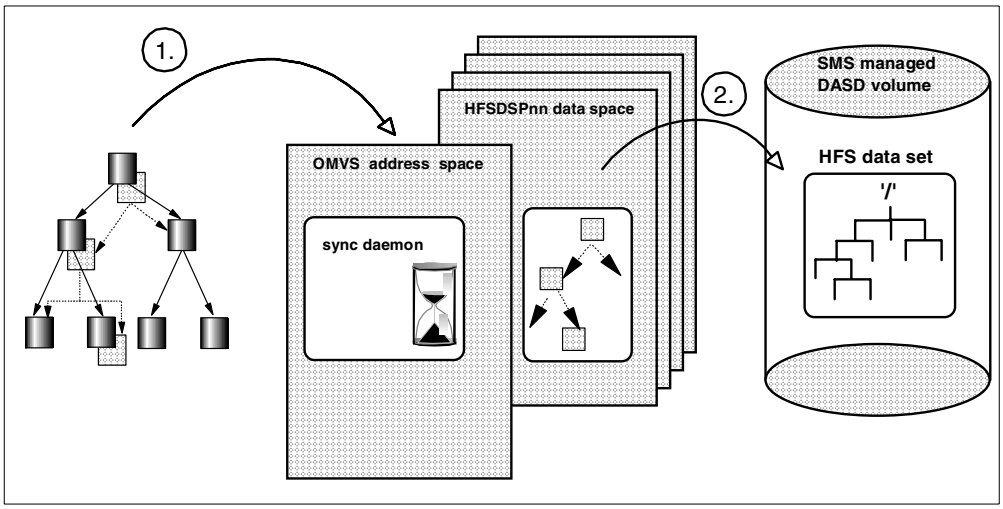


Figure 11. Overview of sync daemon processing

Now, every change inside an HFS will be cached in storage first (1). The cached buffers which have been modified will be marked for processing by the *sync daemon*. The sync daemon will be called, for example, every 60 seconds to write the modified cached data to DASD (2). This 60 second period is referred to as the *sync interval.* Changes are visible immediately. For example, they don't have to be written out to be seen.

The sync interval can be defined in parmlib member BPXPRMxx through the FILESYSTYPE statement for all HFS data sets in an OS/390 UNIX system or at mount time for a specific HFS data set.

Refer to "UNIX System Services Assembler Callable Services: sync (BPX1SYN) — Schedule File System Updates" in the book *OS/390 V2R9.0 UNIX System Services Assembler Callable Services,*SC28-1899. Also refer to "PFS Operations vfs_sync" in the book *OS/390 V2R9.0 UNIX System Services File System Interface Reference,* SC28-1909. Both functions write to disk (harden) all the changed data in a buffer cache for files in a mounted file system.

In DFSMS 1.5, four new data spaces (associated with address space OMVS) are used to cache the index and data buffers. The data space SYSBMFDS associated with the address space SYSBMAS will no longer be used to cache HFS index or data pages. The new data space names are HFSDSP01, HFSDSP02, HFSDSP03 and HFSDSP04.

The amount of virtual storage, and the number of fixed pages used for caching, can be influenced or limited by BPXPRMxx PARMLIB settings, too.

The benefits of sync daemon processing are:

- Improved performance.
- Less expensive index shadow writes.
- If the system crashes between two sync processes, you still have a consistent HFS data set structure — including index, directory and data files — which was written at the previous sync.

  However, you will lose the data residing in the cache that had been modified since the previous sync interval. This means you could miss newly created files, and newly removed files will still be present.

  Application programs can issue an fsync() call which makes a synchronous I/O request to the OMVS kernel, then all the other metadata pages on the same HFS data set will be synchronously hardened.

  See 8.1.1, “HFS deferred writes” on page 185 for further details regarding the sync process.

APAR OW43822

Users affected: Users of DFSMS HFS 1.5.

Problem description:

When an HFS file system encounters an Out of Space condition during SYNC processing, producing message IGW022S in the LOGREC and on the console, and then gives a return code of 00000085x with a reason code of 5Bxx0E37x for any further file function requests, it is necessary to extend the file system. The file system can be extended via the confighfs shell command, but it then still requires the file system to be unmounted and remounted. Also, any file system updates which occurred since the last successful SYNC are lost.

If the HFS SYNC processing is not able to acquire enough available pages in the file system to create a copy of all the index pages which must be modified to commit the file system changes which have been made since the last SYNC, the index updates are lost, losing the file system changes. The file system is flagged as in error and this flag can only be reset by unmounting the file system.

Problem conclusion:

The HFS SYNC process has been modified to capture and maintain the state of the index update in memory when the Out of Space failure occurs. If confighfs is used to successfully extend the file system (for example, confighfs -x <amount> <filesys>) and the extent was large enough to accommodate the pages required to perform the index update, confighfs reinvokes the SYNC process to complete its update and then resets the HFS Out of Space Error flag. It will not be necessary to unmount and remount the file system to use it further. Once the error flag is reset, all file system functions will work properly again.

If the extend amount is not large enough to provide the amount of space required to complete the SYNC process, confighfs will issue the following response:

```
Inadequate space added to HFS. At least another <nn> tracks required.
```

Note: This fix *only* applies for the case when the IGW022S message indicates an `Error Loc: EXTEND` value. If it indicates an `Error Loc: ARPN` value (that is, the failure occurred in the middle of ARPN), it will continue to function as it did prior to this update by resetting back to the last SYNC point and requiring an unmount followed by a mount.

1.3 New enhancements

There were significant enhancements to HFS functions in DFSMS/MVS 1.5 and in OS/390 2.9. Most recently, PTFs have become available to allow the allocation and use of non-SMS managed HFS data sets.

1.3.1 DFSMS/MVS 1.5 enhancements

DFSMS/MVS 1.5, which was made available in March 1999, introduces the following major enhancements to HFS:

- Performance improvements

  It improves performance by implementing deferred I/O and new cache management. See 1.2.4, “Sync process” on page 11, and Chapter 8, “Optimizing HFS performance” on page 185, for more details.

- Multi-volume support

  An HFS can now be a multi–volume data set. This support simplifies the implementation of certain OS/390 USS applications, such as Domino/390. See Chapter 4, “Allocating and mounting HFS data sets” on page 77 for more information.

- Mount integrity improvement (write protection)

  An HFS data set maintains an internal time stamp. The time stamp is set when a system mounts an HFS in read/write mode and is verified by every sync or index read. If the time stamp no longer matches the mount time then no update will be done to the HFS. Chapter 7, “Sharing and serialization for HFS data sets” on page 167 explains this in more detail.

- New PARM parameters in BPXPRMxx PARMLIB member

  You can set three new optional parameters in the FILESYSTYPE HFS statement of the BPXPRMxx PARMLIB member:

  - The **SYNCDEFAULT(t)** parameter specifies the number of seconds, **t**, to be used as a default for the sync daemon interval.
  - The **VIRTUAL(max)** parameter specifies the maximum amount of virtual storage (in megabytes) that HFS data and metadata buffers should use. The default is 50% of real storage available to the system at HFS initialization time.
  - The **FIXED(min)** parameter specifies the amount of virtual storage (in megabytes) that is fixed at HFS initialization time and remains fixed even if HFS activity drops to zero.

  See 3.3.1, “BPXPRMxx options” on page 56, and 3.3.2, “TSO MOUNT command” on page 63 for more information.

- New confighfs shell command

  Interactive shell users can use the confighfs shell command to query the HFS limits set in the BPXPRMxx PARMLIB member and change them dynamically. Superuser authority is required to change the limits. See 3.3.5, “confighfs shell command” on page 68 for more information.

1.3.2 OS/390 2.9 enhancements

OS/390 2.9 provides the capability for sysplex users to access data throughout the file hierarchy. Before OS/390 2.9, users logged onto one system in a sysplex had write access only to the file systems associated with their own system. See also Chapter 7, “Sharing and serialization for HFS data sets” on page 167 for more details.

1.3.3 Non-SMS HFS data sets

In response to customer demand for simpler maintenance procedures and easier system cloning, HFS data sets no longer need to be SMS managed. As a result of this new support, SMS management is now optional when allocating a HFS data set. This support is shipped as PTFs for DFSMS/MVS 1.4 and 1.5 and the PTFs affect these system components:

- DFSMSdfp
- DFSMSdss
- DFSMShsm
- TSO/E
- ISPF
- MVS Base Control Program (Allocation component)

There are restrictions associated with this support. See 3.2, “Non-SMS considerations” on page 51 for additional information regarding the support of non-SMS managed HFS data sets.

Note

In addition to this redbook, please refer to Appendix D, “Related publications” on page 309, for a selection of titles, which contain information about HFS.

1.3.4 DFSORT support of HFS data sets

The PTF for APAR PQ35111 is a small programming enhancement for DFSORT release 14. Among many other functions, it provides the ability for DFSORT to use HFS files for both sort input and output.

Chapter 2. Introduction to OS/390 UNIX System Services

The term *OS/390 UNIX System Services* (USS) and its abbreviation *OS/390 UNIX* are new names for what was previously known as *OpenEdition* in earlier levels of OS/390 and MVS.

The system programmer or data administrator must manage the hierarchically organized data that the system and its users will use. This overall structure of data is called the *hierarchical file system* (HFS). It consists of the root file system and all the file systems that are added to it.

This chapter provides background information about USS to help the system programmer or data administrator understand the logical view of HFS.

2.1 Required USS operation level to use HFS

UNIX System Services provides three levels of operation. To use HFS, you must set up *the full-function mode* of USS.

This topic summarizes each operation level and the tasks for setup.

The OMVS parameter in the IEASYSxx PARMLIB member lets you specify one or more BPXPRMxx PARMLIB members that are used to specify the initial PARMLIB settings for the kernel of UNIX system services.

Activation of kernel services is available in three levels:

- Minimum mode
- Sockets-only mode
- Full-function mode

Note: As of OS/390 Version 2 Release 7, activation of kernel services is available in two levels: minimum mode or full-function mode.

2.1.1 Minimum mode

If you do not specify OMVS= in the IEASYSxx PARMLIB member or if you specify OMVS=DEFAULT, then kernel services start in minimum mode when the system is IPLed. The minimum mode is intended for installations that do not plan to use the kernel services.

In minimum mode:

- Many services are available, but some functions, such as TCP/IP sockets, which require other system customization, may not work.
- TCP/IP sockets (AF_INET) are not available.
- A Temporary File System (TFS) is used. A TFS is an in-storage file system, therefore, no physical DASD data set needs to exist or to be mounted.

  The TFS is initialized and primed with a minimum set of files and directories. Any data written to this file system is not written to DASD.

  The TFS is initialized with these directories and files:

  - / (root directory)
  - /bin directory

- /etc directory
- /tmp directory
- /dev directory
- /dev/null file

There are no executables in the TFS (that is, you will not find the shell and utilities). Do not attempt to install *UNIX System Services Application Services* in the TFS, since no data will be written to DASD.

This is the minimum requirement for applications to be able to use kernel services.

In the minimum mode, you do not need to install or customize SMS, or customize the security product, to work with kernel services.

If the default is set up, you do not need to define users who want to run an application that uses APIs. For example, you would not need to assign a UID to an application that wants to use C pthread functions.

2.1.2 Sockets-only mode

For a sockets-only level of kernel services that is more than the minimum, but that does not need SMS or the shell and other UNIX utilities (such as rlogin), you can create a BPXPRMxx PARMLIB member containing some definitions.

The BPXPRMxx PARMLIB member:

- Takes the default minimum mode and uses the TFS.
- Sets up the DOMAIN NAME specifications for the communication server socket connections.

Whether or not you can run in minimum mode when using sockets depends on a number of factors. If you are using only AF_UNIX sockets, you should be able to run in minimum mode. However, if you are using AF_INET sockets, it will depend on which transport provider you are using:

- TCP/IP requires the use of a file system, it cannot be run in minimum mode.
- Other AF_INET providers, such as OESTACK, do not require a file system. Those can be run in minimum mode.

For each transport provider you specify in the FILESYSTYPE or SUBFILESYSTYPE statements in your BPXPRMxx PARMLIB member, you need to check whether a file system is required.

2.1.3 Full-function mode

If an active BPXPRMxx PARMLIB member specifies FILESYSTEM TYPE(HFS), then the kernel services start in the full-function mode when the system is IPLed. The important point to remember is that you have to set up the full-function mode of USS to use HFS.

To use the full-function mode, you need to:

- Set up BPXPRMxx PARMLIB definitions
- Set up the SMS subsystem
- Set up the hierarchical file systems (HFS)
- Install UNIX System Services Application Services

- Set up the security product definitions for OS/390 UNIX
- Set up the users' access and their individual file systems

2.2 UNIX System Services and related address spaces

This topic summarizes the functions of the address spaces which comprise USS.

Figure 12 shows a diagram of USS running under OS/390 after you have completed setting up full-function mode.

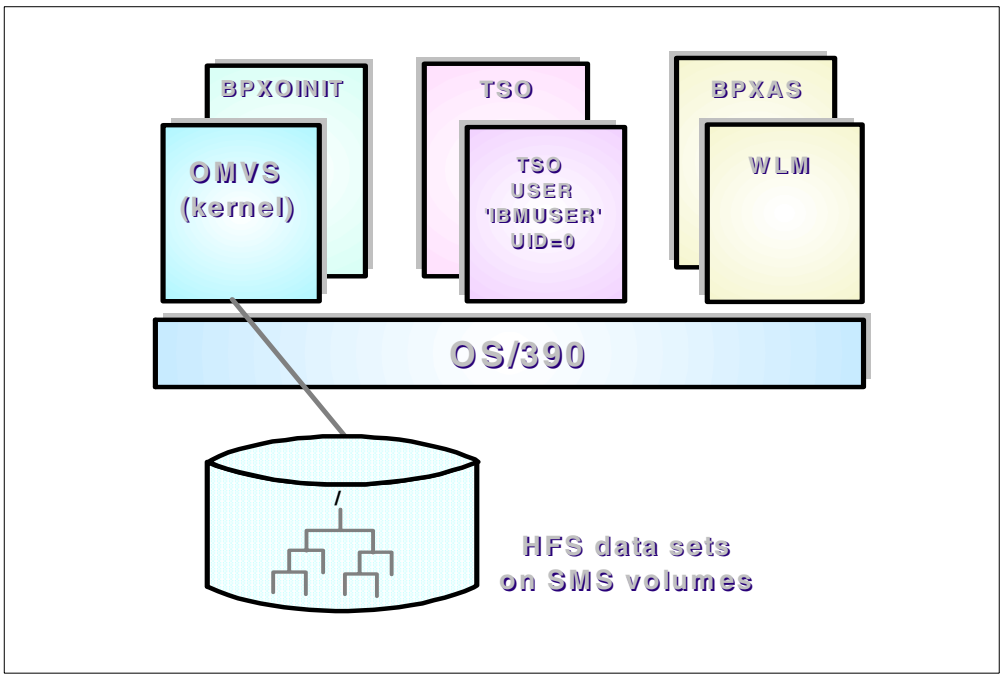


Figure 12. UNIX System Services and related address spaces

The address spaces and functions are described here:

- **OMVS:** OMVS is the address space that initializes the kernel.

  Prior to DFSMS/MVS 1.5, HFS buffer management was processed in the SYSBMAS address space, which was also used by PDSE. Beginning with DFSMS/MVS 1.5, HFS buffer management is processed in OMVS with a new function, the *sync daemon*.

  For more information about the *sync daemon*, please refer to 8.1.1, "HFS deferred writes" on page 185.

- **BPXOINIT:** Beginning with OS/390 1.3, BPXOINIT is the started procedure that runs the initialization process. BPXOINIT is also the job name of the initialization process. Prior to OS/390 1.3, the initialization process was created through an APPC allocate and the job name was OMVSINIT.

  The BPXOINIT address space has two categories of function:

  1. It behaves as PID(1) of a typical UNIX system. This is the parent of /etc/rc, and it inherits orphaned children so that their processes are cleaned up using normal code in the kernel. This task is also the parent of any MVS address space that is dubbed and not created by fork or spawn. Therefore, TSO commands and batch jobs will have a parent PID of 1.

2. Certain functions that the kernel performs need to be able to make normal kernel calls. This address space is used for these activities; for example, mmap() and user ID alias processing.

- **BPXAS:** Prior to OS/390 1.4, APPC/MVS transaction initiators provided address spaces when programs used the fork() or spawn() C functions or callable services. Now, the Workload Manager (WLM) creates the address spaces.

  The BPXAS will now be used to create the WLM-managed address spaces. When using the WLM, you do not need to do any tuning or issue any commands.
- **WLM:** When programs issue the fork() or spawn() C functions or OS/390 callable services, WLM provides a new address space to the BPXAS address space.
- **TSO and TSO user:** A TSO user who has superuser authority (uid=0) is needed to install and customize USS environments.

2.3 UNIX file system overview

This topic provides background information about the UNIX file system, the *hierarchical file system*.

2.3.1 UNIX file system structure

The hierarchical file system consists of:

- Hierarchical file system (HFS) files, which contain data or programs. A file containing a load module or shell script or REXX program is called an executable file. Files are stored in directories.
- Directories, which contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with the root directory at the top and the branches at the bottom. The root is the first directory for the file system at the peak of the tree and is designated by a slash (/). Figure 13 is an example of a UNIX root directory, containing a typical set of subdirectories.
- Additional local or remote file systems, which are mounted on directories of the root file system or of additional file systems.

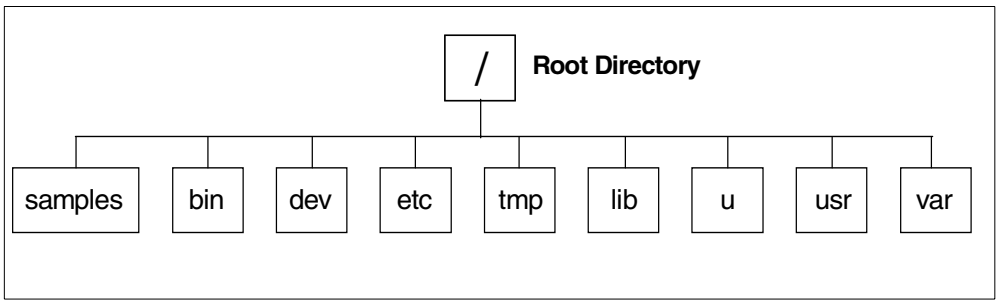


Figure 13. Example of the UNIX file system directory structure

To the OS/390 system, the file hierarchy is a collection of HFS data sets. Each HFS data set is a mountable file system. The root file system is the first file

system mounted. Subsequent file systems can be logically mounted on a directory within the root file system or on a directory within any mounted file system.

Except for the direction of the slashes, the hierarchical file system is similar to a Disk Operating System (DOS), Windows, or an OS/2 file system.

When you install OS/390 USS, you define the root file system using the ROOT statement in the BPXPRMxx member of SYS1.PARMLIB.

The BPXPRMxx PARMLIB member contains the parameters that control OS/390 UNIX System Services processing and the file system. The system uses these values when initializing UNIX System Services.

Important Information

A BPXPRMxx member is provided in SYS1.PARMLIB. Within this member, the root HFS in the single file system is referred to by:

```
ROOT FILESYSTEM('OMVS.OS390R7.ROOT' )
            TYPE(HFS)
            MODE(RDWR)
```

For a detailed description of the BPXPRMxx parameters, see *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890, and *OS/390 V2R9.0 MVS Initialization and Tuning Reference,* SC28-1752.

The root file system is the starting point for the overall HFS file structure. It contains the root directory and any related HFS files or subdirectories.

As of DFSMS/MVS 1.5, HFS data sets can span up to 59 volumes, with up to 255 total extents for all volumes, and up to 123 extents per volume, if secondary extents are specified in the allocation. JCL is typically used to create new HFS data sets by specifying DSNTYPE=HFS on the DD statement. For more information about the new multi-volume support, see 4.1.1, “Multi-volume allocation considerations” on page 78. In addition, it is not always necessary for a secondary space to be specified for an HFS allocation. More extents can be dynamically added later by using the confighfs command if no secondary space exists for an HFS.

HFS file data is byte-oriented, unlike most MVS data sets, which are record-oriented. The OS/390 shell and utilities typically impose a line orientation on the byte-oriented files. A line is a stream of bytes terminated with a <newline> character. A line terminated by a <newline> character is sometimes referred to as *a record*. So, there is a single <newline> character between every pair of adjacent records. Text files use the <newline> character to delimit lines; binary files do not.

Input and output (I/O) for HFS files is typically performed through the use of a data stream. Despite the differences between them, you can copy HFS files into MVS data sets, and MVS data sets into HFS files, using special TSO commands like OCOPY, OGET and OPUT. For more information about these commands, see *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892.

You can access HFS files and manipulate data from application programs using defined C functions. You can also access HFS files using standard MVS access methods. Refer to *DFSMS/MVS Version 1 Release 5 Using Data Sets,* SC26-4922.

2.3.2 Recommended file system structure

Beginning with OS/390 2.6, IBM recommends that you install all OS/390 elements and features into a consolidated root file system, instead of having separate product-related HFS data sets mounted on respective directories. This makes maintaining and cloning of the file system easier, and it also simplifies the MOUNT statements in the BPXPRMxx PARMLIB member.

In addition, we recommend the root file system is set up so that it does not require frequent changes. To this end, we recommend maintaining separate file systems for each of these directories:

- **/etc** — This directory contains customization data. Keeping this information in its own mountable file system will allow you to propagate customized data when migrating from release to release. In addition, you can be assured that IBM products will only create directories under this directory, therefore, not affecting your customization.
- **/var** — This directory contains dynamic data used internally by products and by elements and features of OS/390. Any files or directories needed are created during execution of code. An example of this is caching data. In addition, you can be assured that IBM products will only create directories or files when code is executed.
- **/tmp** — This directory contains temporary data used by products and applications. IBM products will not install any files into this directory. You have the option of mounting a Temporary File System (TFS) on **/tmp**.
- **/u** — Name each user home directory **/u/userid**, where **userid** is the user ID in lowercase.
- **/dev** — This directory contains character-special files that are used when logging into the OMVS shell environment and also during c89 processing. Beginning with OS/390 2.7, /dev is shipped as an empty file and the necessary files are created when the system is IPL'd, and on a demand basis.

Figure 14 shows a single HFS containing a typical UNIX directory structure, with separate HFS data sets mounted at appropriate directory mount points. Note that /u and /tmp may be set up differently than normal HFS data sets to make use of the TFS and the AUTOMOUNT features of OS/390 UNIX System Services.

For more information about how to allocate HFS data sets, see 4.1, “Allocating an HFS” on page 77.

For more information about the AUTOMOUNT feature, see A.2, “Automount facility” on page 248.

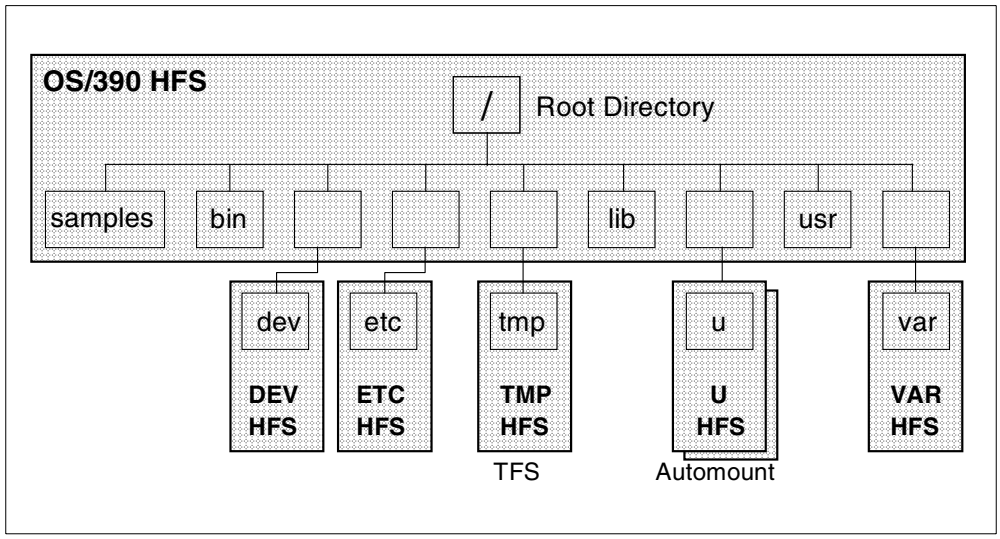


Figure 14. Example of recommended OS/390 HFS structure

Keep system HFS data sets separate from user HFS data sets by creating separate SMS storage groups to segregate the HFS data sets or by allocating non-SMS managed HFS data sets on separate volumes. For more information about SMS considerations, see 3.1, “SMS considerations” on page 33.

Important Information

A BPXPRMxx member is provided in SYS1.PARMLIB. Within this member, the HFS mounted on ’/etc’ is referred to by:

```
MOUNT FILESYSTEM('OMVS.OS390R7.ETC')
            MOUNTPOINT('/etc')
            TYPE(HFS)
            MODE(RDWR)
```

For a detailed description of the BPXPRMxx parameters, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890, and *OS/390 V2R9.0 MVS Initialization and Tuning Reference*, SC28-1752.

For users, you should logically mount other file systems (HFS data sets) on the root file system, and have your users place their directories and files in the mounted file systems. You may want to allocate one large data set for each organization within a company. Separate mountable user file systems offer several advantages:

- They improve storage management because the system administrator need only allocate data sets large enough to accommodate the needs of individual users.
- They enable failure isolation because the system administrator can unmount the user file system that caused an error without affecting other users' data or causing OS/390 USS to fail.
- They relieve the contention for system resources that could occur by having multiple users in a single file system.

The number of mounted file systems is only limited by the number of allocations that can be done in the address space. You can logically mount these user file systems to an empty directory (mount point) in the root file system using the TSO MOUNT command. Use the TSO UNMOUNT command to unmount a file system.

For more information about MOUNT operations, see 4.2, “Mounting an HFS” on page 91.

2.4 HFS and UNIX System Services security

UNIX System Services (USS) is tightly integrated with the OS/390 Security Server (RACF).

The system programmer or data administrator must know the concepts of UNIX and USS security to manage the individual hierarchical file system (HFS) files (not data sets). It is especially important to understand the concept of a *superuser*.

This topic provides some background information about UNIX and USS security. It also provides information about HFS security.

2.4.1 UNIX file security overview

On UNIX systems, each user needs an account that is made up of a user name and a password. UNIX uses the /etc/passwd file to keep track of user names and encrypted passwords for every user on the system (this file is not used in OS/390 USS). Internally, the UNIX operating system uses a numeric ID to refer to a user, so in addition to user name and password, the /etc/passwd file also contains a numeric ID called the user identifier or UID.

UNIX operating systems differ, but generally these UIDs are unsigned 16-bit numbers, ranging from zero to 65,535. OS/390 UNIX System Services supports UID numbers up to 2,147,483,647.

2.4.2 UNIX users and superuser

There is no convention for assigning UID numbers to users other than 0 (zero), which has special significance in that it indicates a superuser. Superusers have special powers and privileges under UNIX, so care must be exercised as to who can gain access to a UID of 0 (zero). If the OS/390 Security Server (RACF) is installed, it offers some features that can be activated to gain more control over superusers. OS/390 Security Server also has capabilities to allow superuser-type access for some specific services for specific users.

The UID is the actual number that the operating system uses to identify the user. User names are provided for convenience, as an easy way for us to remember our sign-on to the UNIX system. If two users are assigned the same UID, UNIX views them as the same user, even if they have different user names and passwords. Two users with the same UID can freely read and write each other′ s files and can kill each other′ s processes.

Note: Assigning the same UID to multiple users is generally not recommended.

2.4.3 UNIX groups

UNIX systems also use the concept of *groups*, where you group together many users who need to access a set of common files, directories, or devices. Like user names and UIDs, groups have both group names and group identification numbers (GIDs). Each user belongs to a primary group that is stored in the /etc/passwd file on UNIX systems (this file is not used in OS/390 USS). OS/390 UNIX System Services supports GID numbers up to 2,147,483,647.

There is no convention for assigning GID numbers, and unlike UIDs, number 0 (zero) has no special significance.

A user's UID and GID values can be displayed using the UNIX id command. Figure 15 shows an id command used to display UID and GID information for a user STYRES2 whose current connect group is ITSOSJ.

```
STYRES2 @ SC64:/>id
uid=67157(STYRES2) gid=2(ITSOSJ)
```

Figure 15. Sample id command

The terms in Figure 15 have the following meanings:

67157: UID

STYRES2: User name associated with UID 67157

2: GID

ITSOSJ: Group name associated with GID 2

Along with the user name, encrypted password, UID, and GID, the /etc/passwd file also contains:

- The user's full name
- The user's home directory (the directory where users generally store their own files and directories)
- The file name of the shell program (also called a shell script) that is executed when the user initially logs in.

In UNIX systems, general information about each file, such as the file size and last modification date, is stored with the files in the file system. Included with this general information is security information about the file, such as:

- The owner (UID)
- The group (GID)
- Unix access controls (permission bits)

File security information can be displayed using a UNIX command such as `ls -l` (`ls` = list file and directory attributes; `-l` = displays permissions, links, owner, group, size, time stamps and name). Figure 16 shows an `ls -l` command to display information about a file called mytext, and the result of that command.

```
STYRES2 @ SC64:/u/styres2>ls -l mytext
-rwxr--r--   1 STYRES2  ITSOSJ 2617 Jun 15 22:13 mytext
```

Figure 16. Sample ls -l command

The terms in Figure 16 have the following meanings:

-rwxr--r--: Permission bits

1: Number of links to the file

STYRES2: Name of the file owner

ITSOSJ: Name of the group that owns the file

2617: Size of the file represented in bytes

Jun 15 22:13: Date and time the file was last changed

mytext: Name of the file

2.4.3.1 Permission bits

All UNIX files have three types of permissions:

- Read (displayed as r)
- Write (displayed as w)
- Execute (displayed as x)

For every UNIX file, read, write and execute (rwx) permissions are maintained for three different types of file user:

- The file owner
- The group that owns the file
- All other users

The permission bits are stored as three octal numbers (3 bits for each type of file user) totalling nine bits. When displayed by commands, such as `ls -l`, a ten-character field is shown, consisting of nine for permission, preceded by one for file type (see the example in Figure 16).

Permission bit structure: The structure of the ten-character field is **tfffgggooo**, where:

t The type of file or directory. Valid values are:

- **-** File
- **c** Character special file
- **d** Directory
- **l** Symbolic link
- **p** FIFO special file

fff The OWNER permissions, as explained in Table 2.

ggg The GROUP permissions, as explained in Table 2.

ooo The OTHER (or sometimes referred to as WORLD permissions), as explained in Table 2.

Table 2. File access type and permission bits

Pos.	Char.	Access Type	Permission for File	Permission for Directory
1	r	Read	Permission to read or print the contents.	Permission to read, but not search, the contents.
2	w	Write	Permission to change, add to, or delete from the contents.	Permission to change, add, or delete directory entries.
3	x	Execute or Search	Permission to run the file. This permission is used for executable files. For OWNER, the third position can also be displayed as: -**s** to indicate an executable file with set-user-ID set. -**S** to indicate a non-executable file with set-user-ID set. For GROUP, the third position can also be displayed as: -**s** to indicate an executable file with set-group-ID set. -**S** to indicate a non-executable file with set-group-ID set. For OTHER, the third position can also be displayed as: -**t** to indicate an executable file with sticky bit set. -**T** to indicate a non-executable file with sticky bit set.	Permission to search the directory.
any	-	No access		

Octal value bits: There are many places in UNIX where permission bits are also displayed as a representation of their octal value. When shown this way, a single digit is used to represent an rwx setting.

The meaning associated with the single digit is:

0 No access (---)

1 Execute-only access (--x)

2 Write-only access (-w-)

3 Write and execute (-wx)

4 Read-only access (r--)

5 Read and execute access (r-x)

6 Read and write access (rw-)

7 Read, write, and execute access (rwx)

Permission bit examples: remembering that each file always has permission bits for owner, group and other, it is usual to see three-digit numbers representing the permission bits of a file. For example, some typical file permission settings are:

666 `Owner('6'='rw-') group('6'='rw-') other('6'='rw-')`

700 `Owner('7'='rwx') group('0'='---') other('0'='---')`

755 `Owner('7'='rwx') group('5'='r-x') other('5'='r-x')`

777 `Owner('7'='rwx') group('7'='rwx') other('7'='rwx')`

User settings: A user may set permission bits for any combination at any level of access. For example, if a user wanted to have read, write, and execute access to one of his/her own files, but not allow access to anyone else, the permission bits would be set to 700, which the `ls -l` command would display as -rwx------.

2.4.3.2 Default permissions set by OS/390

When you first create a file or directory, OS/390 sets default read, write, and execute (rwx) permissions.

Table 3 shows the default permissions set by the system.

Table 3. Default permission bits

Using	To Create a	Default Permissions
mkdir shell command	Directory	owner=rwx group=rwx other=rwx In octal form: 777
MKDIR TSO command	Directory	owner=rwx group=r-x other=r-x In octal form: 755
JCL with no PATHDISP specified	Directory or File	owner=--- group=--- other=--- In octal form: 000
ISPF editor, OEDIT and oedit command	File	owner=rwx group=--- other=--- In octal form: 700
vi editor	File	owner=rw- group=rw- other=rw- In octal form: 666
ed editor	File	owner=rw- group=rw- other=rw- In octal form: 666

Using	To Create a	Default Permissions
Redirections (>)	File	owner=rw- group=rw- other=rw- In octal form: 666
cp command	File	Sets the output file permissions to the input file permissions.
OCOPY command	File	Permission bits for a new file are specified with the ALLOCATE command, using the PATHMODE keyword, prior to entering the OCOPY command. If the PATHMODE keyword is omitted, the default is: owner=--- group=--- other=--- In octal form: 000
OPUT or OPUTX command	File	For a text file: owner=rw- group=--- other=--- In octal form: 600 For a binary file: owner=rwx group=--- other=--- In octal form: 700

If you want to control the permissions that a program can set when it creates a file or directory, you can set a file mode creation mask using the umask command. See *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892 for more details on the umask command.

2.4.4 UNIX file security with the OS/390 Security Server (RACF)

With OS/390 UNIX System Services, the concept of user accounts is the same as for any UNIX system, but the method of storing this account information is different. RACF, when used with OS/390 UNIX System Services, integrates the UNIX account information with the existing MVS account and system information to provide a central secure database in which to store all security information.

Each UNIX user must have a UID and GID assigned to them. These UIDs and GIDs are used by UNIX System Services to control or check access to files and processes. UNIX System Services security functions are implemented in RACF partially as modifications to existing RACF functions, and partially as new RACF functions. The security functions provided include user validation, file access checking, and privileged user checking.

UNIX System Services users are defined with RACF commands. When a job starts or a user logs on, the user ID and password are verified by existing MVS

and RACF functions. When an address space requests a UNIX System Services function for the first time, RACF:

1. Verifies that the user is a valid UNIX System Services user. The user has been assigned a UID.
2. Verifies that the user's current connect group is a valid UNIX System Services group. The current connect group has been assigned a GID.
3. Initializes the control blocks needed for subsequent security checks.

When the first initialization is done, file security checking is started. RACF:

1. Checks whether a started task has *trusted* or *privileged* attributes. If a started task has those attributes, it is treated like superuser.
2. Checks whether the user is superuser (uid=0).
3. Checks whether the UID of the user is UID of owner of the file. The owner permission is used by user. If owner permission bit is 0, user's file access is denied.
4. Checks whether the GID of the user is GID of owner of the file. The group permission is used by user. If group permission bit is 0, user's file access is denied.
5. Checks whether the other permission is not 0. If other permission is 0, user's file access is denied.

Figure 17 shows the flow chart for checking file security.

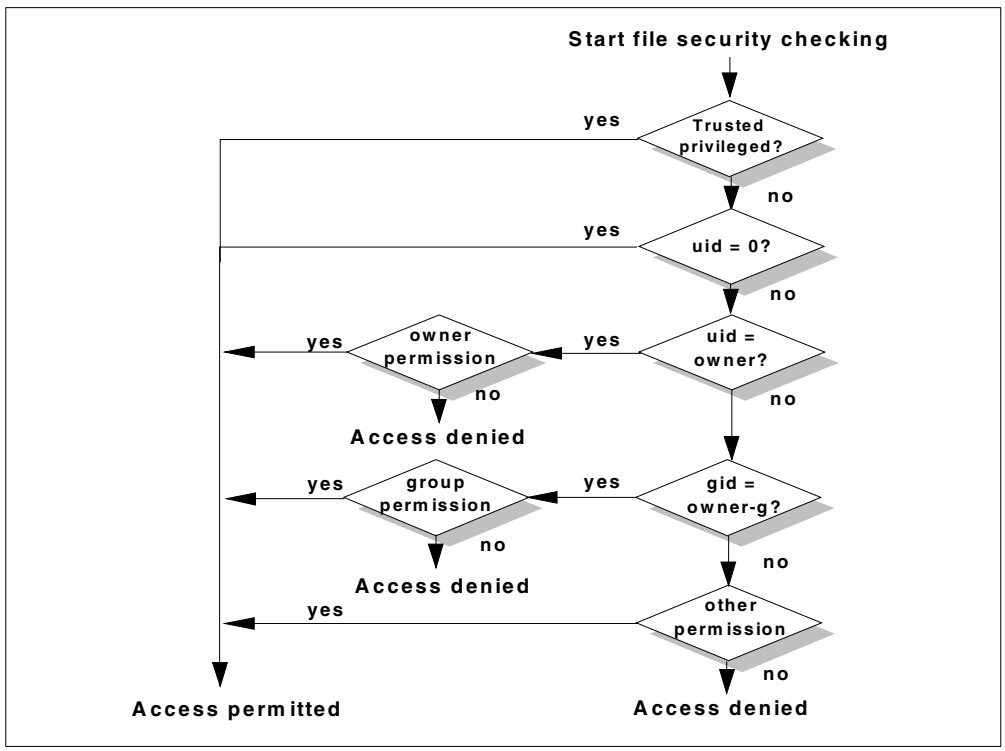


Figure 17. Flow chart for checking file security

Note: RACF does not perform security level checking for a started task that has the RACF *privileged* and *trusted* attribute. A started task with the *privileged* or *trusted* attribute is treated like a superuser (uid=0).

The RACF database contains, among other things, user profiles and group profiles. Associated with these user and group profiles is a new segment called the OMVS segment.

For a user, the OMVS segment contains the user identifier (UID), initial directory path name (HOME), and initial program to execute after logon (PROGRAM). To display this information, the OMVS keyword should be appended to the `RACF LU TSO` command. For example, to display the OMVS segment for user STYRES2, the following command should be used:

```
 ------------------------------------------------------------------------
                      ISPF Command Shell
                      Enter TSO or Workstation commands below:

===> lu styres2 omvs noracf

USER=STYRES2

OMVS INFORMATION
----------------
UID= 0000067157
HOME= /u/styres2
PROGRAM= /bin/sh
```

The NORACF keyword prevents the standard non-OMVS related information from being displayed.

For a group, the OMVS segment contains the group identifier (GID). To display this information, the OMVS keyword should be appended to the `RACF LG TSO` command. For example, to display the OMVS segment for group OMVSGRP, the following command should be used:

```
 ------------------------------------------------------------------------
                      ISPF Command Shell

                      Enter TSO or Workstation commands below:
===> lg omvsgrp omvs noracf



OMVS INFORMATION
----------------
GID= 0000000001
```

Refer to *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890, and *OS/390 V2R8.0Security Server Command Language Reference,* SC28-1919, if you need to define the RACF environment required to install and run UNIX System Services.

Chapter 3. HFS externals

This chapter provides HFS related information about non-SMS and SMS managed HFS data sets, BPXPRMxx PARMLIB member statements, new USS commands, and HFS requirements and restrictions.

3.1 SMS considerations

Storage Management Subsystem (SMS) volume selection is handled in the same manner for HFS data sets as for other system managed data sets in an SMS environment.

We do not explain how to implement SMS nor how to manage system managed data sets in general in this redbook. However, we will describe some common SMS terms for those who are not familiar with SMS. Refer to 3.1.1, “Basic SMS terms” on page 33.

See *DFSMS/MVS DFSMSdfp Storage Administration Reference,*SC26-4920, for additional information regarding SMS, and see the redbook *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customization,* SG24-5178, for a sample SMS implementation.

Information regarding HFS data sets and SMS is provided in subsequent 3.1.3, “Data class” on page 36 through 3.1.7, “ACS routines” on page 46.

Note: Non-SMS managed HFS data sets are supported in DFSMS/MVS 1.4 and 1.5 after PTFs have been installed. However, at the present time, these non-SMS managed HFS data sets must be cataloged at mount time and must be single volume data sets.

3.1.1 Basic SMS terms

SMS, introduced by MVS/DFP 3.1, provides a range of data and space management functions. SMS improves storage space use, controls external storage centrally, and lets you manage storage growth. SMS makes it easier to convert to new device types and takes advantage of what available hardware can do. With SMS, you can move toward system-managed storage.

SMS manages an installation's storage according to the currently active storage management policy. Through the Interactive Storage Management Facility ISMF, you define an installation storage management policy in an SMS configuration. An SMS configuration contains:

- Base configuration information
- Classes and groups
- Automatic class selection (ACS) routines
- Volume selection

Base configuration
The base configuration identifies the systems that the SMS configuration manages. These systems constitute an SMS complex:

- **SMS complex**: A system or a collection of systems that share a common configuration including a common ACDS and a common communication data

set (COMMDS) pair. The SMS configuration supports up to 32 system names, system group names, or both.

The base configuration also contains installation defaults.

You can define more than one control data set, but only one at a time controls SMS. Each control data set defined for SMS is called a source control data set (SCDS). The control data set that is in effect at a given time is the active control data set (ACDS).

An SMS configuration can contain multiple constructs of each type. Data sets managed by SMS are called system-managed. Each system-managed data set or object must reside in a storage group. The system-managed data sets must have a storage class, and might also have a management class and a data class. You can assign the same name to various SMS classes and a storage group. For example, a data class and a storage class can have the same name.

Classes and groups

SMS classes and groups are lists of traits and characteristics that are associated with or assigned to data sets, objects and volumes. An SMS configuration can contain the following five types of classes and groups:

- **Storage group (SG)** allows you to define a list of volumes and manage them as if they were one large, single volume. SMS applies the properties you assign to a storage group to all the volumes within the storage group.
- **Management class (MC)** allows you to define different levels of migration, backup and retention services. Through management class, you can associate a level of service with a data set or object that is independent of the physical location of the data set or object. Also, you can identify an object characteristic that might trigger a class transition.
- **Storage class (SC)** allows you to define different levels of performance and availability services. Through storage class, you can separate the level of service for a data set or object from physical device characteristics. You can also separate the level of service for an object with different storage classes used to place objects at various levels of the storage hierarchy.
- **Data class (DC)** allows you to define allocation defaults. Through the data class, you can simplify and standardize the allocation of new data sets.
- **Aggregate Group** allows you to define groups of data sets for the purpose of backing up or recovering all data sets in a group in a single operation.

Automatic class selection routines

ACS routines determine the SMS classes and storage groups for data sets and objects. You can also use ACS routines to control the transition of data sets and objects to and from SMS management.

Volume Selection

SMS classifies all volumes in the selected storage groups into four volume categories during *volume selection* process:

- **Primary:** Volumes that meet all the specified criteria in the storage class in addition to the volumes being online and below the maximum space used threshold. Both the volume status and storage group status are enabled. Volume selection starts from this list.

- **Secondary:** Volumes that do not meet all the criteria for primary volumes. If there are no primary volumes, SMS selects from the secondary volumes.
- **Tertiary:** Volumes are marked tertiary if the number of volumes in the storage group is less than the number of volumes requested. If there are no secondary volumes available, SMS selects from the tertiary candidates.
- **Rejected:** Volumes that do not meet the required specifications (ACCESSIBILITY = CONTINUOUS, AVAILABILITY = STANDARD or CONTINUOUS, ENABLED or QUIESCED, ONLINE...). These volumes are marked rejected and are not candidates for selection.

Refer to 6.6, "Understanding Volume Selection" in *DFSMS/MVS DFSMSdfp Storage Administration Reference,* SC26-4920, for detailed information on volume selection.

3.1.2 Defining SMS constructs for HFS data sets

In the DFSMS environment, you use SMS classes and groups to set service requirements, performance goals, and data definition models for your installation. You use the ISMF to create the appropriate classes and groups, and ACS routines to assign them to data according to your installation's policies.

Figure 18 provides an overview of allocation in an SMS environment.

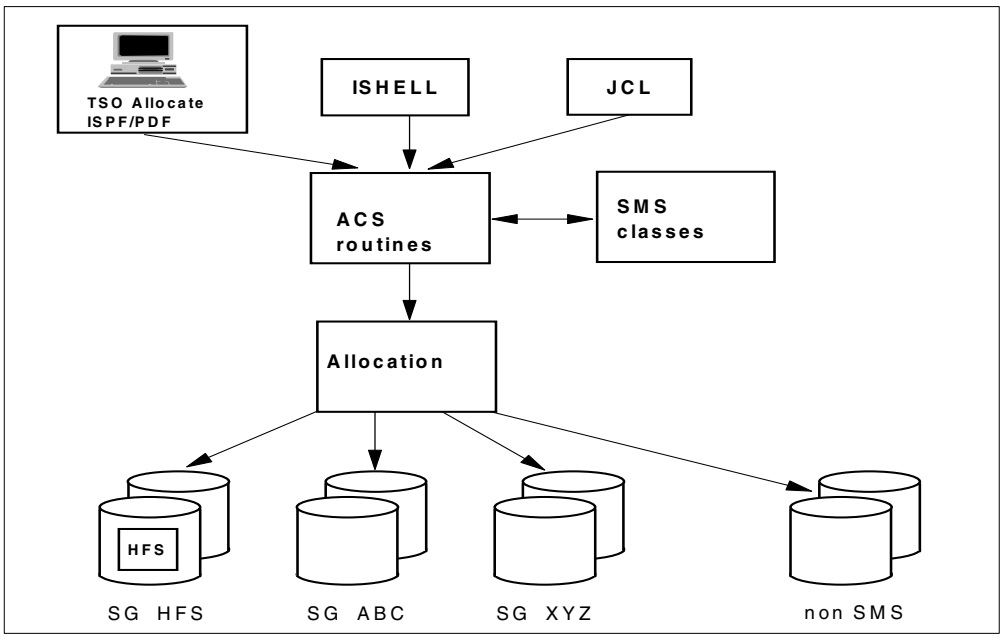


Figure 18. Overview of allocation

You can specify a value of HFS as the Data Set Name Type attribute on the ISMF Data Class Define and Data Class Alter panels. The Data Class List panel indicates whether selected data sets are HFS data sets.

HFS data sets should have a separate data class. You must assign a valid storage class in the SC ACS routine. Otherwise, the HFS data set will be allocated as a non-SMS data set. HFS data sets should also be placed in a pool (DASD) storage group.

ACS routines allow the HFS value if the &DSNTYPE read-only variable is also provided.

3.1.3 Data class

A data class (DC) defines what the data looks like and contains attributes that correspond to parameters that can be coded on JCL DD statements, TSO ALLOCATE commands, or requests for dynamic allocation. It is a collection of allocation and space attributes used to create a data set. You define data classes for data sets that have similar attributes. When end users allocate a data set and refer to a DC, SMS allocates the data set using the attribute values of its associated DC.

You can also specify space attributes such as *Avgrec*, *Avg Value, Primary* and *Secondary* for space to be allocated in this DC. Therefore, you do not need to specify track and cylinder space requests on your allocation request.

- The AVGREC field shows whether space is allocated in bytes (U), kilobytes (K), or megabytes (M).
- The AVG VALUE field shows the length, in bytes, of each record.
- The PRIMARY and SECONDARY fields show the number of kilobytes, megabytes, or bytes allocated for primary and secondary storage.

You can define a data set organization such as HFS in a data class definition. You can create two DC definitions for HFS data set allocation:

1. A data class definition for single volume allocation, as shown:

   Data Set Name Type: The *Data Set Name Type* field defines the format of data sets created using this data class.

   For example, you can define a new Data Class with the name HFS. The only data class attribute you need to specify is *Data Set Name Type = HFS* to allocated an HFS data set.

```
 DATA CLASS DEFINE              Page 3 of 3
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : HFS

To DEFINE Data Class, Specify:
  Data Set Name Type  . . . . . . HFS  (EXT, HFS, LIB, PDS or blank)
    If Ext  . . . . . . . . . . .      (P=Preferred, R=Required or blank)
    Extended Addressability . . . N    (Y or N)
    Record Access Bias  . . . . .      (S=System, U=User or blank)
  Reuse . . . . . . . . . . . . . N    (Y or N)
  Initial Load  . . . . . . . . . R    (S=Speed, R=Recovery or blank)
  Spanned / Nonspanned  . . . . .      (S=Spanned, N=Nonspanned or blank)
  BWO . . . . . . . . . . . . . .      (TC=TYPECICS, TI=TYPEIMS, NO or blank)
  Log . . . . . . . . . . . . . .      (N=NONE, U=UNDO, A=ALL or blank)
  Logstream Id  . . . . . . . . .
  Space Constraint Relief . . . . N    (Y or N)
    Reduce Space Up To (%)  . . .      (0 to 99 or blank)
```

2. A data class with the name HFSMULTI for multi-volume HFS data set allocation:

   Volume Count: Use the *Volume Count* field to specify the maximum number of volumes that can be used to store the HFS data set. Of course, you also need to specify *Data Set Name Type = HFS* to allocate an HFS (multi-volume) data set.

   Note: When a storage group does not contain enough volumes to satisfy the volume count, all volumes in the storage group are flagged as tertiary.

```
 DATA CLASS DEFINE                 Page 2 of 3
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : HFSMULTI
To DEFINE Data Class, Specify:
  Retpd or Expdt  . . . . . .          (0 to 9999, YYYY/MM/DD or blank)
  Volume Count  . . . . . . . 3        (1 to 59 or blank)
    Add'l Volume Amount . . .          (P=Primary, S=Secondary or blank)
  Imbed . . . . . . . . . . .          (Y, N or blank)
  Replicate . . . . . . . . .          (Y, N or blank)
  CIsize Data . . . . . . . .          (1 to 32768 or blank)
  % Freespace CI  . . . . . .          (0 to 100 or blank)
              CA  . . . . . .          (0 to 100 or blank)
  Shareoptions Xregion  . . .          (1 to 4 or blank)
               Xsystem  . . .          (3, 4 or blank)
  Compaction  . . . . . . . .          (Y, N, T, G or blank)
  Media Interchange
    Media Type  . . . . . . .          (1, 2, 3, 4 or blank)
    Recording Technology  . .          (18, 36, 128 or blank)
```

3.1.4 Storage class

A storage class (SC) is a list of storage objectives and requirements. Each SC represents a list of services that are available to data sets having similar access requirements. A SC does not represent any physical storage, but rather provides the criteria that SMS uses in determining an appropriate location to place a data set.

In general, SMS attempts to select a location that meets or exceeds the specified objective, but SMS does not guarantee response time. You should be careful in specifying any attributes that are not reflected by your current configuration.

3.1.4.1 Defining Performance Objectives

In the performance objectives fields of the Storage Class Define panel, you can request millisecond response (MSR) times and indicate the bias of both direct and sequential access data sets. All of the performance attributes are optional.

If you leave all MSR and bias fields blank (direct and sequential), SMS ignores device performance during volume selection.

The MSR serves two purposes in SMS.

- First, it is used as the performance objective for selecting candidate volumes for new data set placement. During a new data set allocation, SMS searches for a volume that meets or closely matches this objective. If no volume satisfies the objective, then SMS attempts to find a volume that comes closest

to matching it. If more than one MSR is explicitly or implicitly specified, the storage class and associated device MSRs are averaged and compared.

- Second, if the data is placed on a volume attached through an IBM 3990 Storage Control with cache, and cache is enabled for that volume, the MSR is used to determine if caching is mandatory, optional, or should be inhibited for the data set.

3.1.4.2 Defining Availability

The Availability field shows whether this storage class can provide uninterrupted processing after a device failure.

Depending on the value in the availability field, SMS selects (or prefers) simplex (like traditional 3390), array (like RAMAC or ESS) and/or dual copy volumes during SMS volume selection process.

NOPREF is the default. Simplex and array DASD are equally considered for volume selection. Dual copy volumes are not candidates for selection.

3.1.4.3 Defining Accessibility

The storage class accessibility attribute defines the function of the hardware supporting point-in-time copy, using either concurrent copy or virtual concurrent copy.

NOPREF (N) is the Default. Point-in-time copy capability is ignored during volume selection.

3.1.4.4 Defining Guaranteed Space

You can allocate space for single volume and multi-volume data sets before the job step runs by specifying a storage class with a Guaranteed Space attribute. SMS fails the request if sufficient space is not available. You can also use the Guaranteed Space attribute to allocate a data set on specific volumes.

For a multi-volume system-managed data set, primary space is preallocated on all the volumes. The first volume becomes the primary volume. All remaining volumes become candidate volumes with preallocated space.

Refer to 4.1, “Allocating an HFS” on page 77 for further information.

For example, if you specify an SC attribute, such as AVAILABILITY, which does not match any of the installed devices, then the volumes are placed into the secondary volume category and SMS may prefer one DASD volume over another. Therefore, the allocation of the HFS data sets are not spread equally over all volumes in selected storage groups.

If you want to spread allocations across volumes of different devices, or devices with different features, use the following parameter settings in the assigned SC.

Storage Class Parameters:

- MSR = blank
- BIAS = blank
- ACCESSIBILITY = NOPREF
- AVAILABILITY = NOPREF

  Note: Dual copy volumes are not candidates for selection.

- GUARANTEED SPACE = N

Pages 1 and 2 of a sample storage class definition are shown in Figure 19 and Figure 20:

```
 STORAGE CLASS ALTER              Page 1 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : OPENMVS
To ALTER Storage Class, Specify:
  Description ==> OE STORAGE CLASS
              ==>
  Performance Objectives
   Direct Millisecond Response . . . .            (1 to 999 or blank)
   Direct Bias . . . . . .                        (R, W or blank)
   Sequential Millisecond Response . .            (1 to 999 or blank)
   Sequential Bias . . . . . .                    (R, W or blank)
   Initial Access Response Seconds . .            (0 to 9999 or blank)
   Sustained Data Rate (MB/sec)  . . .            (0 to 999 or blank)
  Availability . . . . . . . . . . . N            (C, P ,S or N)
  Accessibility  . . . . . . . . . . N            (C, P ,S or N)
   Backup  . . . . . .                            (Y, N or Blank)
   Versioning  . . . . . .                        (Y, N or Blank)
```

Figure 19. Storage class definitions, Page 1 of 2

```
 STORAGE CLASS ALTER              Page 2 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Class Name  : OPENMVS

To ALTER Storage Class, Specify:

  Guaranteed Space . . . . . N          (Y or N)
  Guaranteed Synchronous Write . . . N  (Y or N)
  CF Cache Set Name  . . . . . .        (up to 8 chars or blank)
  CF Direct Weight . . . . . .          (1 to 11 or blank)
  CF Sequential Weight . . . . . .      (1 to 11 or blank)
```

Figure 20. Storage class definitions, Page 2 of 2

3.1.4.5 Non-SMS storage class

By definition, a data set with a storage class is system-managed. If you want to have non-SMS HFS data sets, we recommend that you define a storage class called NONSMS with the defaults.

You may use this storage class in an allocation request and your storage class ACS routine should exit with a null storage class when the NONSMS storage class has been requested in accordance with your installation standards. This will give you a non-managed HFS data set.

3.1.5 Management class

A management class (MC) is a named list of data set migration, backup and retention attribute values. DFSMShsm uses the attributes of the MC associated

with a data set to manage storage. A management class is optional for system-managed data sets.

The MC definition is divided into several sections. For HFS processing, we will discuss only three sections:

- Defining management class expiration attributes (page 1 of 5)
- Defining management class migration attributes (page 2 of 5)
- Defining management class backup attributes (page 2 of 5)

3.1.5.1 Expiration attributes and retention limit

You use expiration attributes to determine the action for HFS data set expiration and deletion. DFSMShsm deletes expired data sets during automatic space management processing. Expiration attributes are required values that indicate when an HFS data set becomes eligible for expiration.

If all the EXPIRE AFTER DAYS NON-USAGE, EXPIRE AFTER DATE/DAYS and RETENTION LIMIT fields have NOLIMIT as the value, the data sets never expire.

```
 MANAGEMENT CLASS DEFINE                Page 1 of 5
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Management Class Name : HFS

To DEFINE Management Class, Specify:

  Description ==>
              ==>

  Expiration Attributes
    Expire after Days Non-usage  . . NOLIMIT     (1 to 9999 or NOLIMIT)
    Expire after Date/Days . . . . . NOLIMIT     (0 to 9999, yyyy/mm/dd or
                                                 NOLIMIT)

  Retention Limit  . . . . . NOLIMIT     (0 to 9999 or NOLIMIT)
```

Partial Release: Use the *Partial Release* field to specify whether allocated but unused space can be released for data sets in this management class. However, since PARTREL is not supported for HFS data sets, you need to specify PARTIAL RELEASE = N.

See 5.4.1, “Releasing unused space” on page 130 for more information.

3.1.5.2 Migration attributes

Primary Days Non-usage: The *Primary Days Non-usage* attribute represents the minimum number of days that must elapse since the last access before a data set is eligible for normal migration.

Level 1 Days Non-usage: If the *Level 1 Days Non-usage* field has specified NOLIMIT, data sets cannot migrate to Level 2 automatically, but they can do so by command, or remain on Level 1 for an unlimited period.

If you do not want to migrate data sets that belong to a particular management class, specify NONE in the Command or Auto Migrate field. The data sets remain on primary storage until they expire.

Notes:

- From a performance point of view, if you plan to migrate HFS data sets, migrate them only to level 1 (DASD) storage. Recalling an HFS data set that was migrated to tape could adversely affect performance because of the time required to physically mount the volume.
- If your HFS data sets reside on RVA volumes, which are already compressed by more than the factor that DFSMShsm or the CPU can compress, we suggest that you eliminate automatic migration to ML1. You can leave data for longer periods of time on the primary volumes, or save some space, and then migrate directly to ML2.

In this sample, the HFS data sets will be migrated to ML1 volumes after 90 days, but they will not be migrated to ML2 volumes.

```
 MANAGEMENT CLASS DEFINE              Page 2 of 5
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Management Class Name : HFS

To DEFINE Management Class, Specify:

  Partial Release . . . . . N          (Y, C, YI, CI or N)

  Migration Attributes
    Primary Days Non-usage  . . . . 90          (0 to 9999 or blank)
    Level 1 Days Non-usage  . . . . NOLIMIT     (0 to 9999, NOLIMIT or blank)
    Command or Auto Migrate . . . . BOTH        (BOTH, COMMAND or NONE)


  GDG Management Attributes
    # GDG Elements on Primary . . .             (0 to 255 or blank)
    Rolled-off GDS Action . . . . .             (MIGRATE, EXPIRE or blank)
```

3.1.5.3 Backup attributes

Number of Backup Versions: The Number of Backup Versions fields specify the maximum number of backup versions to retain for a data set.

Retain Days Only Backup Versions: The Retain days only Backup Version (Data Set Deleted) field indicates how many days to keep the most recent backup version of a deleted data set, starting from the day DFSMShsm detects it has been deleted.

Retain Days Extra Backup Versions: The Retain days extra Backup versions field indicates how many days to keep backup versions other than the most recent one, starting from the day backups were created. It only applies when more than one backup version exists, and when a data set has low activity.

Admin or User command Backup: The Admin or User command Backup field indicates if both the end user and the storage administrator can issue command backups of the data sets in this management class, if only the storage administrator can, or if neither of them can.

Auto Backup: The Auto Backup field is required, and has a default value of Y. If you specify Y in the Auto Backup field, the remaining fields on this panel are required. Otherwise, the remaining fields are optional.

Backup Copy Technique: The Backup Copy Technique field specifies whether or not concurrent copy should be used during data set backup processing.

We used the default settings for the backup attributes. Depending on your backup policy, you may change the defaults.

```
 MANAGEMENT CLASS DEFINE            Page 3 of 5
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Management Class Name : HFS

To DEFINE Management Class, Specify:
  Backup Attributes
    Backup Frequency  . . . . . . . . 1        (0 to 9999 or blank)
    Number of Backup Vers . . . . . . 2        (1 to 100 or blank)
      (Data Set Exists)
    Number of Backup Vers . . . . . . 1        (0 to 100 or blank)
      (Data Set Deleted)
    Retain days only Backup Ver . . . 60       (1 to 9999, NOLIMIT or blank)
      (Data Set Deleted)
    Retain days extra Backup Vers . . 30       (1 to 9999, NOLIMIT or blank)
    Admin or User command Backup  . . BOTH     (BOTH, ADMIN or NONE)
    Auto Backup . . . . . . . . . . . Y        (Y or N)
    Backup Copy Technique . . . . . . S        (P=Conc Preferred, R=Conc
```

3.1.6 Storage group

Storage groups (SG) represent the physical storage managed by SMS. This storage can be collections of DASD volumes, volumes in tape libraries, optical devices, or virtual input/output (VIO) storage. For HFS data set allocations, you can only use an storage group of DASD volumes. A storage group, used together with storage classes, separates the logical requirements for accessing data from the physical requirements to store the data. You can set up storage group attributes to specify how the system should manage the storage group.

3.1.6.1 Storage group considerations

- You use the SG ACS routine to assign a new data set to a storage group. You can assign multiple candidate storage groups. In this case, the system chooses a specific storage group from your list. Storage group definitions are not apparent to users.
- Two storage groups cannot share a DASD volume. You must define an entire volume to a single pool-type storage group. Also, a data set can only reside in one pool-type storage group. A data set can span volumes within a single pool-type storage group, but it cannot span volumes belonging to several pool-type storage groups.
- Although not required, we recommend that you define pool-type storage groups so that they only contain devices of the same geometry. The device geometry is the track size and number of tracks per cylinder for the device.

  By defining pool storage groups so that the device geometry is the same for all volumes in the storage group, you can ensure that volumes of the same

geometry are available when multi-volume data sets need to extend to new volumes.

For example, if you have 3380 and 3390 devices, you should define at least two storage groups: one containing 3380 devices, and another containing 3390 devices.

Since 3390 devices in 3380 track compatibility mode are geometrically the same as 3380 devices, you can combine these devices in a single storage group. Because the 3390 devices are in 3380 track compatibility mode, the access methods see them as 3380 devices.

- Although you should separate devices according to geometry, you do not need to separate them according to capacity. For example, you can combine all models of the 3390 into a single SG. The only effect that the different capacities have is on volume thresholds. See *SML Managing Storage Groups,* SC26-3125 for information on selecting appropriate threshold levels.
- Devices of the same geometry can have different performance characteristics. These devices coexist in the same storage group, and enhanced volume selection for SMS manages data set placement accordingly. With enhanced volume selection, even devices with vastly different performance characteristics can reside in the same storage group.
- If a data set has a management class that specifies automatic backup or migration, you must direct the data set to a storage group that is eligible to be processed for automatic backup or migration.

See *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920, and *SML Managing Storage Groups,* SC26-3125, for more information about planning and implementing storage groups.

There is no requirement to create your own storage group for HFS data sets. If you have already implemented a storage management policy, you can use your existing policy to manage HFS data sets.

However, if you have not implemented a storage management policy, we suggest that you create a separate set of storage groups for HFS data sets. Doing this provides benefits such as:

- Causing less impact on performance from other OS/390 applications, for example, to reduce queuing on the UCB.
- Simplifying multi-volume processing.
- Directing the backup processing to the system which has mounted the HFS.
- Separating your USS data (HFS files) from other OS/390 applications.

The storage group *Auto Migrate* and *Auto Backup* parameters specify whether the volumes in this storage group are eligible to be processed automatically. The management class assigned to the data sets residing on the volumes determines whether and how to process the data sets on the volume. In contrast, if you set *Auto Migrate* or *Auto Backup* to NO in the storage group attributes, the volumes in the storage group are not processed and the data sets residing in the storage group are neither migrated nor backed up.

After a storage group has been set up, volumes can be defined for this storage group.

3.1.6.2 Auto Migrate

Use the AUTO MIGRATE field to specify whether data sets on volumes in this storage group can be moved to migration level 1 DASD or migration level 2 tape by the primary space management and interval migration functions of DFSMShsm.

During primary space management, which is run daily, DFSMShsm moves data sets until the space allocated on each volume drops to or below the MIGR LOW value, or no more data sets on the volume are eligible to be migrated.

In its hourly interval migration, DFSMShsm moves eligible data sets from each volume with allocation at or above the MIGR HIGH value, until the allocation reaches the MIGR LOW value.

Possible values for the auto migrate attribute are:

Y Data sets are eligible for primary space management migration. If SETSYS INTERVALMIGRATION has been specified in DFSMShsm, the data sets are also eligible for interval migration.

N Data sets are not eligible for automatic migration.

I Data sets are eligible for primary space management and interval migration.

P Data sets are eligible for primary space management but not interval migration.

When AUTO MIGRATE is I, migration is done when the space used exceeds the half way mark between the MIGR HIGH and MIGR LOW thresholds.

3.1.6.3 Auto Backup

Use the *Auto Backup* field to specify whether all the volumes in the SG are eligible for automatic backup. If the volumes are eligible for automatic backup, each data set on the volume will be backed up according to the backup attributes of its management class. If you specify that the volumes are not eligible for automatic backup, the backup attributes for the data sets on those volumes will be ignored. Specify Y if your volumes in this SG are eligible for automatic backup.

3.1.6.4 Backup Sys/Sys Group Name

The *Backup Sys/Sys Group Name* field shows the name of the system or system group where the automatic backup function will be processed. All data sets in the storage group are eligible to be backed up by the system or system group specified.

Note: You should specify the name of the system that has mounted the HFS R/W in the *Backup Sys/Sys Group Name* field. The backup will be performed on the same system which has the HFS currently mounted in R/W mode. Refer to 5.2, “DFSMShsm backup and migration” on page 115 for more information about HFS and DFSMShsm.

3.1.6.5 Auto Dump

In *Auto Dump*, you specify whether you want automatically to dump all the DASD volumes in this storage group. It is an optional field which ISMF primes with the value N (No).

Note: We suggest that you do not use AUTODUMP Y. This will result in DFSMShsm full volume dump processing, which invokes DFSMSdss physical volume dump. DFSMSdss physical dump does not quiesce mounted HFSs in the dump data set. Although no error indication will be given during dump processing, a subsequent restore may result in a damaged or unusable HFS. Refer to 5.1, "DFSMSdss dump and restore" on page 103 for more information about HFS and DFSMSdss.

3.1.6.6 Allocation/migration Threshold

Use the HIGH and LOW values of *Allocation/migration Threshold* to optimize the use of DASD space in a pool storage group. SMS tries to stay below the HIGH value when choosing a volume on which to allocate a new data set. DFSMShsm uses a threshold to determine whether to run the interval migration function. When utilization meets or exceeds the threshold, DFSMShsm migrates data sets until volume utilization is below the LOW value, or there are no more data sets eligible for migration.

In addition, DFSMShsm uses the LOW threshold during its daily primary space management function. If the amount of allocated space exceeds the LOW threshold, data sets are deleted or migrated until the LOW threshold is met.

If *Auto Migrate* is Y and SETSYS INTERVALMIGRATION has been specified in DFSMShsm, DFSMShsm checks hourly to see if the HIGH value has been reached. If AUTO MIGRATE is I, DFSMShsm checks the space used to see if it exceeds the half way mark between the HIGH and LOW values. If it does, DFSMShsm starts interval migration.

Both the HIGH and LOW values represent the percentage of DASD space to be used. A HIGH value is required and has a default of 85. A LOW value is required if AUTO MIGRATE is Y, I or P. If LOW is specified, it must be less than or equal to HIGH.

3.1.6.7 Guaranteed Backup Frequency

Use the *Guaranteed Backup Frequency* field to specify the maximum number of days that can elapse between incremental data set backups being taken. During this backup period, a backup copy of each data set within the storage group is available. This field is valid only for pool-type storage groups.

You must provide a backup frequency when AUTO BACKUP is set to YES. When AUTO BACKUP is NO, backup frequency is optional.

Specify NOLIMIT if your data sets in the SG are backed up according to management class specifications.

The use of guaranteed backup frequency means that you can recover a volume from incremental backup tapes only. No full volume dump is used and the use of guaranteed backup frequency limits the number of backup tapes that are mounted for recovery. This technique is good for HFS data sets because there is no DFSMSdss physical dump processing involved.

You must specify GBF=1 if you want to get at least one DFSMShsm backup version per day. See 5.2.2, "Backup processing" on page 118 for more information regarding DFSMShsm backup consideration.

3.1.6.8 DEFINE SMS Storage Group Status

Use the *DEFINE SMS Storage Group Status* ISMF panel to designate the relationship or status between storage groups and the systems in a complex. Initially, all of the status fields are set to ENABLE. Refer to *DFSMS/MVS DFSMSdfp Storage Administration Reference,* SC26-4920, for further information about the meanings of the storage group status field.

The following screen shows a sample ISMF panel for POOL STORAGE GROUP DEFINE.

```
 POOL STORAGE GROUP DEFINE
Command ===>

SCDS Name . . . . . : SYS1.SMS.SCDS
Storage Group Name  : HFS
To DEFINE Storage Group, Specify:
 Description ==>
             ==>
 Auto Migrate . . Y  (Y, N, I or P)   Migrate Sys/Sys Group Name . .
 Auto Backup  . . Y  (Y or N)         Backup Sys/Sys Group Name  . .
 Auto Dump  . . . N  (Y or N)         Dump Sys/Sys Group Name  . . .

 Dump Class . . .                     (1 to 8 characters)
 Dump Class . . .                     Dump Class . . .
 Dump Class . . .                     Dump Class . . .

 Allocation/migration Threshold: High . . 85  (1-99)      Low  . . 50  (0-99)
 Guaranteed Backup Frequency  . . . . . . NOLIMIT   (1 to 9999 or NOLIMIT)

 DEFINE SMS Storage Group Status . . .... N   (Y or N)
```

3.1.7 ACS routines

You use ACS routines to assign class and storage group definitions to data sets as well as to HFS data sets.

Through ISMF, you can create and maintain as many as four ACS routines in an SCDS, one for each type of SMS class and one for storage groups.

Figure 21 shows the order in which ACS routines are processed. Data will become system-managed if the storage class routine assigns a storage class to the data set or if a user-specified SC is assigned to the data set. For HFS data sets that are to be system-managed, a storage class must be assigned.

A storage group must be assigned by the ACS routine, because that is the only way to specify a SG.

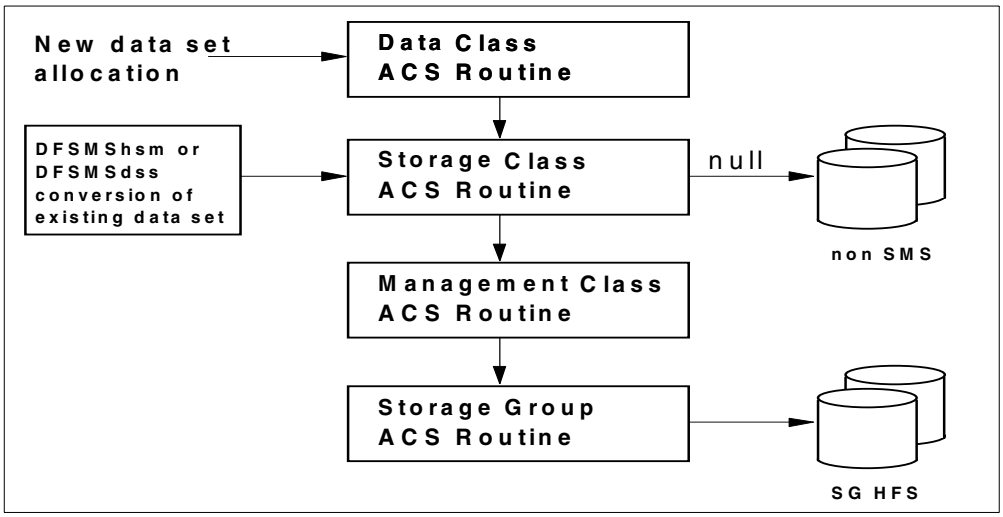


Figure 21. Overview of ACS routines

The ACS language contains a number of read-only variables, which you can use to analyze new data allocations. For example, you can use the read-only variable &DSN to make class and group assignments based on data set name, or &LLQ to make assignments based on the low-level qualifier of the data set or object collection name.

You cannot alter the value of read-only variables. You use the four read-write variables (&DATACLAS, &STORCLAS, &MGMTCLAS and &STORGRP) to assign the class or storage group you determine for the data set, based on the routine you are writing. You may only set a read-write variable in its own ACS routine.

For a detailed description of the ACS language and its variables, see *DFSMS/MVS DFSMSdfp Storage Administration Reference,* SC26-4920.

3.1.7.1 Sample ACS routine definitions for HFS data set allocations

The following are sample ACS routine definitions for HFS data set allocation together with two example batch job streams which allocate HFS data sets.

Note: We extracted only that information from the different ACS routines that is relevant for HFS data set allocations.

In our sample ACS routines, all data sets which meet one of the following conditions will be allocated as HFS data sets:

- Pre-assigned data class of HFS. Refer to DDNAME2 (single volume) on page 50.
- Pre-assigned data class of HFSMULTI. Refer to DDNAME5 (multi-volume) on page 50.
- DSNTYPE=HFS. Refer to DDNAME1 and DDNAME4 on page 50.
- Low level qualifier (&LLQ), starting with HFS. Refer to DDNAME3 and DDNAME6 on page 50.

Data class ACS routine

The DC ACS routine assigns a data class of HFS or HFSMULTI to all data sets that meet one of the following conditions:

- DATACLAS of HFS or HFSMULTI is pre-assigned
- DSNTYPE=HFS is specified
- The LLQ of data set names starts with HFS

```
FILTLIST HFS_DATA_SET INCLUDE(HFS*)


SELECT
  WHEN (&DATACLAS= 'HFS')
    DO
      SET &DATACLAS = 'HFS'
    END
  WHEN (&DATACLAS= 'HFSMULTI' )
    DO
      SET &DATACLAS = 'HFSMULTI'
    END
  WHEN (&DSNTYPE EQ 'HFS')
    DO
      SET &DATACLAS EQ 'HFS'
    END
  WHEN (&LLQ EQ &HFS_DATA_SET)
    DO
     SET &DATACLAS EQ 'HFS'
   END
```

Note: The benefit of explicitly assigning the HFS data class to allocations that have already defined DSNTYPE=HFS is that the HFS data set will always be SMS managed, due to our definitions in the SC ACS routine.

For example, if we remove the part,

```
WHEN (&DSNTYPE EQ 'HFS') DO SET &DATACLAS EQ 'HFS' END
```

in the DC ACS routine, and we try to allocate an HFS data set with a name other than HFS* or OMVS*, (in our sample SMS environment), the SC ACS routine routes the allocation to non-SMS managed volumes, although we have specified DSNTYPE=HFS in the JCL. The allocation will give us a non-SMS managed HFS data sets which may not be what we want.

Storage class ACS routine

All data sets with a high level qualifier (HLQ) starting with OMVS or HFS will be given the OPENMVS storage class. All data sets which have a data class of HFS or HFSMULTI will also be given the OPENMVS storage class.

Note:

1. Ensure that you assign a valid storage class in the storage class ACS routine for your HFS data set allocations. Otherwise, the HFS data set will be allocated as a non-SMS data set.
2. Non-SMS managed HFS data sets are supported. However, these non-SMS managed HFS data sets must be cataloged at mount time and can only be single volume data sets.

```
FILTLIST OMVS     INCLUDE(OMVS.**,HFS.**)
FILTLIST HFSDC    INCLUDE('HFS','HFSMULTI')

    IF &HLQ = &OMVS
      THEN DO
        SET &STORCLAS = 'OPENMVS'
      EXIT
    END

    IF &DATACLAS = &HFSDC
      THEN DO
        SET &STORCLAS = 'OPENMVS'
      EXIT
    END
```

Management class ACS routine

The HFS management class will be assigned to all data sets that have already been assigned the OPENMVS storage class.

```
      IF &STORCLAS EQ 'OPENMVS'
        THEN DO
          SET &MGMTCLAS = 'HFS'
        EXIT
      END
```

Storage group ACS routine

The OPENMVS storage group will be assigned to all data sets that are associated with the OPENMVS storage class.

```
SELECT
    WHEN (&STORCLAS = 'OPENMVS')
    SET &STORGRP = 'OPENMVS'
```

Sample batch job allocations based on the ACS routine definitions

Part 1 of 2 (single-volume HFS data set allocation):

```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* #1  DSNTYPE=HFS                                               */
//*****************************************************************/
//DDNAME1  DD DSN=STYRES1.DATA.SET1,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(TRK,(2,2,1)),
//            DSNTYPE=HFS
//*****************************************************************/
//* #2  DATACLAS=HFS                                              */
//*****************************************************************/
//DDNAME2  DD DSN=STYRES1.DATA.SET2,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(TRK,(2,2,1)),
//            DATACLAS=HFS
//*****************************************************************/
//* #3  DSN LLQ = HFS*                                            */
//*****************************************************************/
//DDNAME3  DD DSN=STYRES1.USER.HFS3,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(TRK,(2,2,1))
```

Part 2 of 2 (multi-volume HFS data set allocation):

```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* #4  MULTIVOLUME  DSNTYPE=HFS                                  */
//*****************************************************************/
//DDNAME4  DD DSN=STYRES1.MULTIVOL.DATA.SET4,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA,3),
//            SPACE=(TRK,(2,2,1)),
//            DSNTYPE=HFS
//*****************************************************************/
//* #5  MULTIVOLUME  DATACLAS=HFSMULTI                            */
//*****************************************************************/
//DDNAME5  DD DSN=STYRES1.MULTIVOL.DATA.SET5,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA,3),
//            SPACE=(TRK,(2,2,1)),
//            DATACLAS=HFSMULTI
//*****************************************************************/
//* #6  MULTIVOLUME   DSN LLQ = HFS*                              */
//*****************************************************************/
//DDNAME6  DD DSN=STYRES1.USER.MULTIVOL.HFS6,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA,3),
//            SPACE=(TRK,(2,2,1))
```

All six allocations in the above samples result in an HFS data set allocation. Table 4 shows which DC was assigned to the HFS data set and whether or not the HFS data set was allocated as a multi-volume HFS data set. All HFS data sets were given the OPENMVS storage class, the HFS management class and assigned to the OPENMVS storage group.

Table 4. Data class assignment results

Allocation	assigned DC	Multi-volume
#1	HFS	No
#2	HFS	No
#3	HFS	No
#4	HFS	Yes
#5*	HFSMULTI	Yes
#6	HFS	Yes

* **Note:** no VOLCOUNT specified in UNIT statement

3.2 Non-SMS considerations

HFS data sets need no longer be SMS managed. As a result of this new support, SMS management is now optional when allocating an HFS data set.

This support is shipped via PTFs for DFSMS/MVS 1.4 and 1.5 and changes DFSMSdfp, DFSMSdss, DFSMShsm as well as TSO/E, ISPF and BCP (Allocation). Please see information APAR II12221 regarding the required maintenance to allocate non-SMS managed HFS data sets. We recommend that you contact your local IBM Support Center.

You may also need to change your current processes and policies to allow non-SMS managed HFS data sets:

- Ensure that your storage class ACS routine allows non-SMS managed HFS data sets.
- Specify VOL=SER=xxxxxx and UNIT parameters in JCL.

The usage of non-SMS managed HFS data sets provides the following benefits:

- It can simplify maintenance procedures and system cloning processing.
- They can reside on the SYSRES volume.

Important information

You can only mount *cataloged* non-SMS managed HFS data sets.

You can only mount *single volume* non-SMS managed HFS data sets.

However, there are some restrictions associated with this support:

- To mount an non-SMS managed HFS data set, it must be cataloged.

- You can create uncataloged HFS data sets, but you will not be able to mount them until they are cataloged.
- You will not be able to mount multi-volume non-SMS managed HFS data sets.

You should not assume that the support for non-SMS HFS data sets is a recommendation to move away from SMS management of HFS data. Exploitation of automated storage management by setting appropriate values in the data, storage and management classes for HFS data sets is still of great value. The support is intended to be use for system data sets and as a convenience for sites who have not yet implemented system-managed storage.

3.2.1 Non-SMS availability management

DFSMShsm manages non-SMS storage using policies that are either system-wide or apply at a volume level. All the data sets on a volume are managed by the same policy, whether for number of days between backups or number of versions retained. This is in contrast to the much more flexible use of policies at a data set level for system-managed data. The differences from processing of SMS managed storage are:

- Uncataloged data sets can be backed up and recovered.
- Users can specify the volume to which a data set is to be recovered.
- Volumes to be dumped or backed up are individually identified to DFSMShsm.
- System-wide settings are held in SYS1.PARMLIB member ARCCMDxx.

Availability management of non-SMS managed storage uses the same techniques for management as those for SMS managed storage. Although the functions performed are similar, the policies for availability management of non-SMS managed storage are specified differently. System-wide policies are set in the DFSMShsm parmlib member ARCCMDxx. Volume-level settings are set in ADDVOL commands which are normally included in ARCCMDxx. Because no storage group or management class definitions are available to define how to manage the volumes and which volumes to manage, there are some parameters that we must set:

- How frequently data sets are to be backed up.

  For SMS managed data sets, you can specify a different backup frequency for each management class. This lets you have as many different backup frequencies as you have management classes. For non-SMS managed storage, you can specify only one backup frequency for all volumes processed by any one processor. You can, however, change the backup frequency of individual non-SMS managed data sets with the (H)ALTERDS command.

  The **FREQUENCY** parameter of the SETSYS command controls the backup frequency for the data sets processed by each processor. For the example, the following command allows DFSMShsm to backup an HFS data set once a day:

  ```
  SETSYS FREQUENCY(1)
  ```

  If you do not specify this parameter on any SETSYS command, the DFSMShsm default is zero, meaning that a data set will be eligible for backup each time that volume backup runs.

- How many versions to keep for each backed up data set.

For non-SMS managed storage, the number of backup versions to retain is a system-wide specification. As with SMS managed storage, depending on the record length used to define the BCDS, DFSMShsm can maintain up to 29, or up to 100, backup versions of any data set. Refer to the *DFSMShsm Implementation and Customization Guide*, SH21-1078, for details.

Within that upper limit, the **VERSIONS** parameter of the SETSYS command controls the number of backup versions to be retained. For the example, the following command specifies a maximum number of five backup versions that DFSMShsm can keep for any one HFS data set:

```
SETSYS VERSIONS(5)
```

If you do not specify this parameter on any SETSYS command, the DFSMShsm default is 2.

The SETSYS VERSIONS parameter is used to specify the number of backup versions for all your non-SMS managed data sets. A DFSMShsm-authorized user, however, can specify the VERSIONS parameter of the ALTERDS command to change that number of backup versions for a specific HFS data set. An unauthorized user can specify the VERSIONS parameter of the HALTERDS command to change that number of backup versions for a specific HFS data set with the same high-level qualifier as the unauthorized user.

- Which volumes are to be **backed up** automatically.

  For non-SMS managed storage, you can identify each individual volume that DFSMShsm backs up automatically by specifying the AUTOBACKUP parameter on the ADDVOL command for that volume.

  For example, you can add following commands to the ARCCMDxx member on the system that will do the backup processing for the non-SMS managed volumes:

  ```
  ADDVOL GP0001 UNIT(3390) PRIMARY(AUTOBACKUP)
  ```

  Add the following commands to the ARCCMDxx member for those processors that should not perform the backup:

  ```
  ADDVOL GP0001 UNIT(3390) PRIMARY(NOAUTOBACKUP)
  ```

- Which volumes are to be **dumped** automatically.

  As with volumes to be backed up, you must identify the non-SMS managed volumes to be dumped automatically.

  Note: We suggest that you do not use AUTODUMP with HFS data sets. This will result in DFSMShsm full volume dump processing, which invokes DFSMSdss physical volume dump. DFSMSdss physical dump does not quiesce mounted HFSs while dumping them. Although no error indication will be given during dump processing, a subsequent restore may result in a damaged or unusable HFS. Refer to 5.1, “DFSMSdss dump and restore” on page 103 for more information about HFS and DFSMSdss.

You also must define a backup cycle if you want to use DFSMShsm incremental backup (for both SMS managed and non-SMS managed data).

The **BACKUP** parameter of the **DEFINE** command specifies a backup cycle that specifies the days on which the processor does automatic backup processing. You can also specify the day to start the cycle so that the cycle does not change

with each re-initialization of DFSMShsm. For example, the command to be added to the ARCCMDxx member in each processor is:

```
DEFINE BACKUP(YYYYYYN CYCLESTARTDATE(95/01/06))
```

The command establishes a 7-day cycle that begins on date 95/01/06 which was a Monday. Thus, backup processing is done on every day except Sunday.

DFSMShsm stores the date you specify with CYCLESTARTDATE as the date the backup cycle began.

3.2.2 Non-SMS space management

We will only describe some basics that are relevant for DFSMShsm migration processing for HFS data sets. For a complete description, please refer to:

- *DFSMShsm Storage Administration Guide*, SH21-1076
- *DFSMShsm Storage Administration Reference*, SH21-1075

DFSMShsm manages non-SMS managed storage on a volume basis. Non-SMS managed volumes that are to be automatically managed by DFSMShsm are known as primary volumes. All the data sets on a volume are managed to the same specification, whether for days-not-used on the primary volume, age for deletion, or permission for automatic migration. In addition, DFSMShsm chooses the volume to receive the recalled data sets based on any volume pools you have defined, as well as the characteristics given to the volumes when you define them.

Differences between processing for SMS managed and non-SMS managed data sets are as follows:

- Two additional management options are defined for non-SMS managed volumes:
  - Deletion (delete by age)

    This is done by expiration specifications in the management class for SMS managed data sets.
  - Retirement (delete if backed up)

  It is extremely unlikely that you would choose either of these options for HFS data sets.
- Level 1 to level 2 migration is controlled on a system-wide basis rather than on a data set basis.
- Target volumes for recall are selected by DFSMShsm rather than by DFSMSdfp ACS routines.
- Volumes are individually identified to DFSMShsm.

To manage non-SMS managed storage, you must define system-wide parameters to define how you want space management to be done. Again, you use the SETSYS, ADDVOL, and DEFINE commands that were described in the previous section. You cannot define storage groups or management classes. In addition, you define parameters to control the following functions:

- Specify the minimum age when data sets should migrate if no age is specified when the volume is defined to DFSMShsm in the ADDVOL command.

DAYS (days) is an optional parameter specifying the number of contiguous days a data set must remain unreferenced before the data set is eligible for migration. DFSMShsm uses this parameter to migrate data sets from primary volumes to migration level 1 volumes. For example:

```
SETSYS DAYS(7)
```

The value specified by the DAYS parameter of the SETSYS command is used only if no value is specified for the MIGRATE parameter of the ADDVOL command. An ADDVOL command with the MIGRATE(0) parameter specified also causes SETSYS DAYS to determine the migration age.

If you do not specify this parameter on any SETSYS command, the DFSMShsm default is one day if DFSMShsm is running in a single processing unit environment or two days if DFSMShsm is running in a multiple processing unit environment.

- Specify when data sets become eligible for level 1 to level 2 migration.

  For non-SMS managed storage, the age at which data sets become eligible for migration from level 1 to level 2 is the same for all non-SMS managed data sets in the system. The MIGRATIONLEVEL1DAYS parameter of the SETSYS command specifies the minimum time that a data set must remain unused on the combination of level 0 and level 1 before it becomes eligible to migrate to migration level 2. For example:

  ```
  SETSYS MIGRATIONLEVEL1DAYS(31)
  ```

  If you do not specify this parameter with any SETSYS command, the DFSMShsm default is 60 calendar days.

- Specify recall characteristics for primary volumes.

  For non-SMS managed data sets, DFSMShsm selects the type of volume to which a data set can be recalled. The characteristics for a volume to be eligible to receive recalled data sets are the same for all non-SMS managed data sets in the system. The RECALL parameter of the SETSYS command specifies the general volume destinations for the recalled data sets.

  See the DFSMShsm books mentioned above for more information.

- Define primary volumes to DFSMShsm.

  The ADDVOL command defines non-SMS managed volumes to DFSMShsm and is not used for SMS managed volumes.

  Here is an example of an ADDVOL command added to the ARCCMDxx member:

  ```
  ADDVOL GP0001 UNIT(3390) PRIMARY(AUTOMIGRATION -
          AUTORECALL MIGRATE(12)) THRESHOLD(95 80)
  ```

  The AUTOMIGRATION subparameter specifies that DFSMShsm is to perform automatic volume space management on the volume.

  The AUTORECALL subparameter specifies that the volume is generally available as a recall volume.

  The MIGRATE subparameters specify the management technique for the volume. In this example, DFSMShsm migrates the data sets that have not been opened for 12 days. If you do not specify a value for the number of days with the MIGRATE subparameter, DFSMShsm uses the value specified with the DAYS parameter of the SETSYS command.

The THRESHOLD parameter in these ADDVOL commands operates in the same way as the THRESHOLD attribute in the storage groups. See 3.1.6, “Storage group” on page 42.

- Define pools of volumes.

  To ensure that recalled non-SMS managed data sets go to the correct volumes, you should organize your non-SMS managed volumes into pools. You can define either pools that accept only specified data set names during recall or pools that accept for recall any data set that migrated from them.

  See the DFSMShsm books mentioned above for more details.

  Finally, you can prevent migration for named data sets by using the SETMIG command. Although it is unlikely that your HFS data sets will be inactive, it is good practice to prevent migration for the data sets that hold the root and key parts of the file system. Migration may be sensible for data sets containing an individual user’s file systems. You can issue the SETMIG command for fully-qualified data set names or for groups of data sets by using a high-level qualifier.

3.3 HFS PARMLIB and command enhancements

The following sections describe the enhancements to the BPXPRMxx PARMLIB member and the new USS confighfs command.

3.3.1 BPXPRMxx options

BPXPRMxx contains the parameters that control the OS/390 UNIX System Services (OS/390 UNIX) environment, the hierarchical file system (HFS), and sockets file systems (AF_UNIX and AF_INET). The system uses these values when initializing the kernel.

OS/390 UNIX services are started during initialization. To specify which BPXPRMxx PARMLIB member to start with, the operator can include OMVS=xx in the reply to the IPL message or can include OMVS=xx in the IEASYSxx PARMLIB member.

If OMVS=xx is not specified in the reply to the IPL message and is not in the IEASYSxx member, or if OMVS=DEFAULT is specified, defaults are used for each parameter and OMVS is initialized with minimal functions. If the operator specifies OMVS=xx in the IPL reply to the message, it overrides the OMVS=xx specified in IEASYSxx.

For additional information about BPXPRMxx options, refer to:

- *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890
- *OS/390 V2R9.0 MVS Initialization and Tuning Reference,* SC28-1752

3.3.1.1 FILESYSTYPE statement

In the FILESYSTYPE statement, specify the TYPE of the file system to be started. BPXPRMxx can contain more than one FILESYSTYPE statement, such as HFS, TFS, or NFS. There are many more types available, but we will cover only the HFS type in this topic. The syntax is:

```
FILESYSTYPE     TYPE(type_name)
                ENTRYPOINT(entry_name)
```

```
PARM('parm')
```

Note

With OS/390 2.8 and later, you can add physical file systems dynamically.

The parameters are:

- **TYPE:** Specifies the name of the file system type that is to control the file system.

  TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

  Specify **HFS** for a hierarchical file system.

- **ENTRYPOINT(entry_name):** Specifies the name of the load module containing the entry point into the file system type.

  ENTRYPOINT is a required parameter. The name is 1 to 8 characters; The system converts the name to uppercase.

  For TYPE(HFS), the ENTRYPOINT must be **GFUAINIT**.

  The system attaches the GFUAINIT load module during OMVS initialization.

- **PARM('parameter'):** Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

  PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both.

  The parameter must be enclosed in single quotes.

  The following parameters are introduced with DFSMS 1.5 to control the *Sync Process* and HFS buffer usage. They are only valid for an HFS with an ENTRYPOINT of GFUAINIT.

  Note: If a syntax error is found in any of these three parameters (SYNCDEFAULT, VIRTUAL, or FIXED), an error message is issued and all three parameters are set to the default values.

  The following parameters are supported by an HFS:

  ```
  SYNCDEFAULT(t),VIRTUAL(max),FIXED(min)
  ```

  - **SYNCDEFAULT(t):** `t` specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the modified metadata that is cached in storage for a file system, is written (hardened) to disk. Setting `t` to 0 indicates that the file system should write metadata synchronously with syscall requests. This means any changes to the metadata will not be deferred, they will be immediately written to disk as it was done in DFSMSdfp 1.4 and older systems.

    Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.

    The maximum value that can be specified for `t` is 65535. Values between 65535 and 99998 are rejected.

A value of 99999 means that the sync daemon is not to be run. Any deferred changes will be written at unmount time or at explicit sync of the file system or during inact processing.

Default: 60 seconds

See 1.2.4, “Sync process” on page 11 for more information regarding the sync process.

- **VIRTUAL(max):** max specifies the maximum amount of virtual storage (in megabytes) that HFS data and metadata buffers should use. You cannot specify a maximum value less than 32M. If less than 32M is specified, an information message is issued, and max is set to 32M. The maximum limit can be changed dynamically by invoking the confighfs OMVS shell command.

  See 3.3.5, “confighfs shell command” on page 68 and OS/390 UNIX System Services Command Reference, SC28-1892 for more information about the confighfs OMVS shell command.

  Note: HFS may temporarily exceed the limit set in max to avoid failure of a file read or write request, but the amount of space used is reduced to the max specification or less as soon as possible.

  Default: 50% of real storage available to the system at HFS initialization time.

- **FIXED(min):** min specifies the amount of virtual storage (in megabytes) that is fixed at HFS initialization time and permanently remains fixed even if HFS activity drops to zero. min must be less than or equal to VIRTUAL(max). The benefit of FIXED is to avoid the overhead of page fixing and unfixing needed for I/O.

  min cannot exceed 50% of the real storage available to the system. If the allowed amount of storage is exceeded, an informational message is issued and min is set to 50% of real storage. The minimum limit can be changed dynamically by invoking the confighfs OMVS shell command.

  For more information about the confighfs OMVS shell command, see:

  - Section 3.3.5, “confighfs shell command” on page 68
  - *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892.

  Note: HFS will continue to fix additional buffers temporarily, as needed, during I/O requests. In addition, HFS will unfix storage and go below FIXED(min) if there is a pageable storage shortage in the system.

  Default: 0

  We recommend that for systems dedicated to HFS usage, the value for FIXED should be equal to the value of VIRTUAL, up to 50% of real storage capacity.

Figure 22 is an example of the FILESYSTYPE statement.

```
FILESYSTYPE TYPE(HFS)            /* Type of file system to start  */
            ENTRYPOINT(GFUAINIT) /* Entry Point of load module    */
            PARM('SYNCDEFAULT(90)')
```

Figure 22. Example of the FILESYSTYPE statement

3.3.1.2 SYSPLEX statement (OS/390 2.9 and above)

The SYSYPLEX statement is:

```
SYSPLEX(YES|NO)
```

For OS/390 UNIX System Services, the SYSPLEX statement specifies whether a system should join the SYSBPX XCF group to share HFS resources across the sysplex. If SYSPLEX(YES) is specified, the system participates in shared HFS. If SYSPLEX(NO) is specified, the system does not participate in shared HFS. If the SYSPLEX statement is not provided, the default is SYSPLEX(NO).

Also, to participate in sharing HFS data sets, the systems must be at OS/390 2.9 level or later. For more information on shared HFS, see “Shared HFS in a Sysplex“ in *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890.

Note: You cannot adjust the SYSPLEX field dynamically. There is no SETOMVS, SET OMVS, or SETOMVS RESET=(xx) capability. To change the value of SYSPLEX, you must re-IPL the system.

Default: NO

3.3.1.3 VERSION statement (OS/390 2.9 and above)

The VERSION statement is:

```
VERSION('nnnn')
```

The VERSION statement applies only to systems that are exploiting shared HFS. VERSION allows multiple releases and service levels of the binaries to coexist and participate in sharing HFS data sets. A directory with the value nnnn specified on VERSION is dynamically created at system initialization under the sysplex root and is used as a mount point for the version HFS. This directory, however, is only dynamically created if the sysplex root HFS is mounted read/write.

Note: nnnn is a case-sensitive character string no greater than 8 characters in length. It indicates a specific instance of the version HFS. The most appropriate values for nnnn are the name of the target zone, &SYSR1, or another qualifier meaningful to the system programmer. For example, if the system is at OS/390 2.9, you could specify REL9 for VERSION.

When SYSPLEX(YES) is specified, you must also specify the VERSION parameter.

The VERSION value is substituted in the content of symbolic links that contain $VERSION. For scenarios describing the use of the version HFS, see “Shared HFS in a Sysplex“ in OS/390 V2R9.0 UNIX System Services Planning, SC28-1890.

When testing or changing to a new maintenance level (PTF level), the VERSION value can be changed dynamically by using the SETOMVS command:

```
SETOMVS VERSION='string'
```

You can also change the settings of this parameter via SET OMVS=(xx) and SETOMVS RESET=(xx) parmlib specifications.

Note: We do not recommend changing VERSION dynamically if you have any users logged on or running applications; Replacing the system files for these users may be disruptive.

3.3.1.4 MOUNT statement

Specifies a file system that OS/390 UNIX is logically to mount as the root file system or another file system. The MOUNT statement is optional; The BPXPRMxx member can contain one or more MOUNT statements.

```
MOUNT      FILESYSTEM('fsname') or DDNAME(ddname)
           TYPE(type_name)
           MOUNTPOINT('pathname')
           MODE(access)
           PARM('parameter')
           SETUID|NOSETUID
           AUTOMOVE|NOAUTOMOVE
           SYSNAME(sysname)
```

The parameters are:

- **FILESYSTEM('file_system_name'):** The name of the file system. The name must be unique in the system.

  Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 44 characters; The characters can be in uppercase, lowercase, or both. The name must be enclosed in single quotation marks. An HFS data set name must conform to the rules for MVS data set names.

- **DDNAME(ddname):** The ddname on the JCL DD statement that defines the file system. To use the DDNAME parameter, a DD statement for the HFS dataset containing the mountable file system should be placed in the OMVS cataloged procedure.

  Either FILESYSTEM or DDNAME is required; Do not specify both. The name is 1 to 8 characters; the system converts the ddname to uppercase.

- **TYPE(type_name):** Specifies the name of a file system type identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement.

  TYPE is a required parameter. The name is 1 to 8 characters; The system converts the name to uppercase.

- **MOUNTPOINT('pathname'):** Specifies the pathname of the directory onto which the file system is to be mounted.

  Mount point restrictions are:

  - The mount point must be a directory.
  - Any files in the directory are not accessible while the file system is mounted.
  - Only one mount can be active at any time for a mount point.

- A file system can be mounted at only one directory at any time.

MOUNTPOINT is required. The pathname is up to 1023 characters long; The characters can be in uppercase, lowercase, or both. The pathname must be enclosed in single quotation marks.

- **MODE(access):** Specifies access to the mounted file system by all users:
  - READ: Users can only read the file system being mounted.
  - RDWR: Users can read and write in the file system being mounted.

  Default: RDWR
- **PARM('parameter'):** Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

  PARM is an optional parameter. The parameter is up to 1024 characters long; The characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotation marks.

  The following parameters are introduced with DFSMS/MVS 1.5 to control the *Sync Process* (refer to 1.2.4, “Sync process” on page 11) and *Write Protection* (refer to 7.2, “Sharing considerations (OS/390 2.8 and below)” on page 174). They are only valid for an HFS with an ENTRYPOINT of GFUAINIT.

  ```
  PARM('SYNC(t),NOWRITEPROTECT')
  ```

  - **SYNC(t):** `t` is a numeric value that specifies the number of seconds that should be used to override the sync interval default for this file system during a specific mount. If SYNC is not specified at mount time, then the sync interval default value will be used (a value of 60 seconds). The same rules apply to the argument to the SYNC keyword at mount time as apply to the argument of the SYNCDEFAULT keyword at HFS initialization time. For more information on the SYNCDEFAULT keyword, see *OS/390 UNIX System Services Planning,* SC28-1890.
  - **NOWRITEPROTECT:** The HFS has a write protection mechanism that adds some overhead to HFS processing. This overhead can be avoided by turning off the write protection by specifying NOWRITEPROTECT in the PARM field of the MOUNT command. See 7.2, “Sharing considerations (OS/390 2.8 and below)” on page 174 for more details.
- **SETUID|NOSETUID:** SETUID specifies that the setuid() and setgid() mode bit on an executable file will be supported.

  NOSETUID specifies that the setuid() and setgid() mode bit on an executable file will not be supported. The UID or GID will not be changed when the program is executed and the APF and Program Control extended attributes are not honored. The entire HFS is uncontrolled.

  Default: SETUID
- **AUTOMOVE | NOAUTOMOVE:** These are valid for OS/390 2.9 and later releases.

  The AUTOMOVE and NOAUTOMOVE parameters apply only in a sysplex where systems are participating in shared HFS. The AUTOMOVE and NOAUTOMOVE parameters indicate what happens if the system that owns a file system goes down. AUTOMOVE indicates that ownership of the file system automatically changes to another system participating in sysplex HFS sharing. NOAUTOMOVE indicates that ownership of the file system is not

moved if the owning system goes down; As a result, the file system becomes inaccessible.

For file systems that are mostly used by Distributed File System (DFS) clients, consider specifying NOAUTOMOVE on the MOUNT statement. By doing so, the file systems will not change ownership if the system is suddenly recycled, and they will be available for automatic re-export by DFS. This is recommended, because a file system can only be exported by the DFS server on the system that owns the file system. Once a file system has been exported by DFS, it cannot be moved until it has been unexported from DFS. When recovering from system outages, you need to weigh sysplex availability against availability to the DFS clients. When an owning system recycles and a DFS-exported file system has been taken over by one of the other systems, DFS cannot automatically re-export that file system. The file system will have to be moved from its current owner back to the original DFS system— the one that has just been recycled— and then exported again.

Default: AUTOMOVE

- **SYSNAME(sysname):** This parameter is valid for OS/390 2.9 and later releases.

  For systems participating in sysplex HFS sharing, SYSNAME specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. This system must be IPLed with SYSPLEX(YES).

  Default: The name of the system, if IPLed with SYSPLEX(YES), that the mount is processed on.

  Note: In OS/390 2.9 and later, to ensure that the root is always available, use the defaults for SYSNAME and AUTOMOVE.

  For additional information, see "MOUNT" in *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890.

Figure 23 is an example of the MOUNT statement.

```
MOUNT FILESYSTEM('OMVS.STYRES1.HFS3')
      MOUNTPOINT('/u/guts')
      TYPE(HFS)  MODE(RDWR)
      PARM('SYNC(120),NOWRITEPROTECT')
```

Figure 23. Example of MOUNT statement

3.3.1.5 ROOT statement

Specifies a file system that OS/390 UNIX is logically to mount as the root file system.

```
ROOT       FILESYSTEM('fsname') or DDNAME(ddname)
           TYPE(type_name)
           MODE(access)
           PARM('parameter')
           SETUID|NOSETUID
           AUTOMOVE|NOAUTOMOVE
           SYSNAME(sysname)
```

The parameters are the same as for the mount statement. Refer to 3.3.1.4, “MOUNT statement” on page 60 for parameter explanations.

Figure 24 is an example of the ROOT statement.

```
ROOT     FILESYSTEM('HFS.OS390R7.&SYSNAME..&SYSR1..ROOT')
         TYPE(HFS)              /* TYPE OF FILE SYSTEM           */
         MODE(RDWR)             /* (OPTIONAL) CAN BE READ OR RDWR. */
         PARM('SYNC(30)')
```

Figure 24. Example of ROOT statement

3.3.2 TSO MOUNT command

For hierarchical file systems, you can use the MOUNT command to logically mount, or add, a mountable file system to the file system hierarchy. You can unmount any mounted file system using the UNMOUNT command.

Note: A mount user must have UID 0 or at least have READ access to the BPX.SUPERUSER FACILITY class.

Syntax:

```
MOUNT FILESYSTEM(file_system_name)
          MOUNTPOINT(pathname)
          TYPE(file_system_type)
          MODE(RDWR|READ)
          PARM('parameter_string')
          SETUID|NOSETUID
          WAIT|NOWAIT
          SECURITY|NOSECURITY
          AUTOMOVE|NOAUTOMOVE
          SYSNAME(sysname)
```

The syntax of some of the following parameters has already been given for the MOUNT statement. Refer to 3.3.1.4, “MOUNT statement” on page 60 for these parameter explanations.

The parameters are:

- **FILESYSTEM(file_system_name):** Specifies the name of the file system to be added to the file system hierarchy.

  file_system_name

  For the hierarchical file system (HFS), this is the fully qualified name of the MVS HFS data set that contains the file system.

  The file system name specified must be unique among previously mounted file systems. The file system name supplied is changed to all uppercase characters. You can enclose it in single quotes, but they are not required.

  If FILESYSTEM("'file_system_name'") is specified, the file system name will not be translated to uppercase.

- **MOUNTPOINT(pathname):** See explanation on page 60.
- **TYPE(file_system_type):** See explanation on page 60.
- **PARM('parameter')**
  - **SYNC(t)**
  - **NOWRITEPROTECT**
- **SETUID|NOSETUID**

- **WAIT|NOWAIT:** Specifies whether to wait for an asynchronous mount to complete before returning.
  - **WAIT:** Specifies that MOUNT is to wait for the mount to complete before returning. WAIT is the default.
  - **NOWAIT:** Specifies that if the file system cannot be mounted immediately (for example, a network mount must be done), then the command will return with a return code indicating that an asynchronous mount is in progress.
- **SECURITY|NOSECURITY:** Specifies whether security checks are to be enforced for files in this file system.
  - **SECURITY:** Specifies that normal security checking will be done. SECURITY is the default.
  - **NOSECURITY:** Specifies that security checking will not be enforced for files in this file system. A user may access or change any file or directory in any way.

  Security auditing will still be performed if the installation is auditing successes. The SETUID, SETGID, APF, and Program Control attributes may be turned on in files in this file system, but they will not be honored while it is mounted with NOSECURITY.
- **AUTOMOVE | NOAUTOMOVE**
- **SYSNAME(sysname)**

Figure 25 is an example of the TSO MOUNT command.

```
MOUNT FILESYSTEM('NIGELR2.TEST.HFS')
      MOUNTPOINT('/u/guts')
      TYPE(HFS)  MODE(RDWR)
      PARM('SYNC(120),NOWRITEPROTECT')
      NOAUTOMOVE
      SYSNAME(SC65)
```

Figure 25. Example of TSO MOUNT command

Refer to 4.2.2, “TSO MOUNT and UNMOUNT commands” on page 93, and *OS/390 V2R9.0 UNIX System Services Command Reference,* SC28-1892, for more information on the TSO MOUNT command.

3.3.3 TSO UNMOUNT command

The UNMOUNT command removes a file system from the file system hierarchy. The alias for this command is UMOUNT.

Note: An UNMOUNT user must have UID 0 or at least have READ access to the BPX.SUPERUSER FACILITY class.

The UNMOUNT command format is as follows:

```
UNMOUNT FILESYSTEM(file_system_name)
             DRAIN | FORCE | IMMEDIATE | NORMAL |
             REMOUNT(RDWR | READ) | RESET
```

The parameters are:

- **FILESYSTEM(file_system_name):** Specifies the name of the file system to be removed from the file system. The name supplied is changed to all uppercase characters. This operand is required.

  file_system_name

  For the hierarchical file system (HFS), this is the fully qualified name of the MVS HFS data set that contains the file system. The file system name supplied is changed to all uppercase characters.

  Specify the name of the file system exactly as it was specified when the file system was originally mounted. You can enclose it in single quotes, but they are not required. If FILESYSTEM('"file_system_name"') is specified, the file system name will not be translated to uppercase.

- **DRAIN:** Specifies that an unmount drain request is to be made. The system will wait for all use of the file system to be 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 <ATTN> (or <PA1>) key does not terminate the command.

- **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 system are saved, if possible. If the data changes to the files cannot be saved, the unmount request continues and data is lost.

  Note: An UNMOUNT IMMEDIATE request must be issued before you can request an UNMOUNT FORCE of a file system. Otherwise, UNMOUNT FORCE fails.

- **IMMEDIATE:** 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 system are saved. If the data changes to files cannot be saved, the unmount request fails.

- **NORMAL:** Specifies that if no user is accessing any of the files in the specified file system, the system processes the unmount request. Otherwise, the system rejects the unmount request. **This is the default**.

- **REMOUNT(RDWR|READ):** Specifies that the specified file system be remounted, changing its mount mode. REMOUNT takes an optional argument of RDRW or READ. If you specify either argument, the file system is remounted in that mode if it is not already in that mode. If you specify REMOUNT without any arguments, the mount mode is changed from RDWR to READ, or READ to RDWR.

- **RESET:** A reset request stops a previous UNMOUNT DRAIN request.

Figure 26 is an example of the TSO UNMOUNT command.

```
UNMOUNT FILESYSTEM('OMVS.STYRES1.HFS5')
```

Figure 26. Example of TSO UNMOUNT command

Refer to 4.2.2, “TSO MOUNT and UNMOUNT commands” on page 93, and *OS/390 V2R9.0 UNIX System Services Command Reference,* SC28-1892, for more information on the TSO UNMOUNT command.

3.3.4 SETOMVS system command

Use the SETOMVS command to change dynamically the options that OS/390 UNIX System Services currently is using. These options are originally set in the BPXPRMxx parmlib member at the time of initially program loading (IPL'ing) the system. For more information on the BPXPRMxx parmlib member, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890 or 3.3.1, “BPXPRMxx options” on page 56.

Note: in this chapter, we only cover the options that are related to HFS processing, for example the new SETOMVS EXTENSIONS for shared HFS data sets in a sysplex. For a complete description of SETOMVS command, see *OS/390 V2R9.0 MVS Systems Commands*, GC28-1781.

Parameters AUTOMOVE=YES|NO, FILESYS=filesys, FILESYSTEM=filesystem, FROMSYS=sysname, MOUNTPOINT=mountpoint, SYSNAME=sysname|*, and VERSION='nnnn', which are described in this section, are parameters that are used in a sysplex environment where systems are exploiting shared HFS. For more information on shared HFS in a sysplex, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890 or Chapter 7, “Sharing and serialization for HFS data sets” on page 167.

```
SETOMVS VERSION='string'
SETOMVS  FILESYS
           ,FILESYSTEM=filesystem
           ,AUTOMOVE=YES|NO
           ,SYSNAME=sysname|*   or
SETOMVS  FILESYS
           ,FILESYSTEM=filesystem
           ,AUTOMOVE=YES|NO     or
SETOMVS  FILESYS
           ,FILESYSTEM=filesystem
           ,SYSNAME=sysname|*   or
SETOMVS  FILESYS
           ,MOUNTPOINT=mountpoin
           ,AUTOMOVE=YES|NO
           ,SYSNAME=sysname|*   or
SETOMVS  FILESYS
           ,MOUNTPOINT=mountpoin
           ,AUTOMOVE=YES|NO     or
SETOMVS  FILESYS
           ,MOUNTPOINT=moun poin
           ,SYSNAME=sysname|*   or
SETOMVS  FILESYS
           ,FROMSYS=sysname
           ,SYSNAME=sysname|*
```

Note: FILESYSTEM, MOUNTPOINT, and FROMSYS are mutually exclusive parameters. When you specify FILESYS, you must supply only one of these three parameters.

- **AUTOMOVE=YES|NO:** The AUTOMOVE | NOAUTOMOVE parameters apply only in a sysplex where systems are participating in shared HFS. The AUTOMOVE and NOAUTOMOVE parameters indicate what happens if the

system that owns a file system goes down. AUTOMOVE indicates that ownership of the file system automatically changes to another system participating in shared HFS. NOAUTOMOVE indicates that ownership of the file system is not moved if the owning system goes down; As a result, the file system becomes inaccessible. AUTOMOVE is the default.

Note: AUTOMOVE is not allowed when moving multiple file systems. Also, in OS/390 2.9 and later, to ensure that the root is always available use the default for AUTOMOVE.

- **FILESYS:** In a sysplex environment, this parameter alerts the parser that commands which change mount attributes are forthcoming. For examples on the use of this parameter when making move or change requests, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890.
  - **FILESYSTEM=filesystem:** In a sysplex environment, FILESYSTEM is the 45 character alphanumeric field that denotes the name of the file system to be changed or moved. This file system name may be in the following form: `'OMVS.USER.JOE'`. FILESYSTEM, MOUNTPOINT, and FROMSYS are mutually exclusive parameters.

    For examples on the use of this parameter when making move or change requests, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890.
  - **FROMSYS=sysname:** In a sysplex environment, this parameter indicates the system where all the file systems will be moved from. The file systems will be moved to the system identified by the sysname keyword. FILESYSTEM, MOUNTPOINT, and FROMSYS are mutually exclusive parameters.
  - **MOUNTPOINT=mountpoint:** In a sysplex environment, MOUNTPOINT is the mountpoint specification. For example:

    '/usr/d1'

    It is case sensitive. This is the mountpoint where the filesystem is mounted. If specified, the file system associated with this mountpoint will be moved or changed. FILESYSTEM, MOUNTPOINT, and FROMSYS are mutually exclusive parameters.

    For examples on the use of this parameter when making move or change requests, see *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890.
- **SYSNAME=sysname|*:** *sysname* is the one to eight character alphanumeric name of a system participating in shared HFS. This system must be IPLed with SYSPLEX(YES). *sysname* specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. If * (asterisk) is specified, it represents any other randomly selected system taking part in shared HFS. The asterisk specification is not available with the FROMSYS parameter.

  For examples of the use of this parameter when making move or change requests, see "Shared HFS in a Sysplex" in *OS/390 UNIX System Services Planning*.
- **VERSION = 'nnnn':** Please see 3.3.1.3, “VERSION statement (OS/390 2.9 and above)” on page 59 for more information regarding the VERSION statement.

In the example in Figure 27, system SC64 will become the owner of file system 'NIGELR2.TEST.HFS'.

```
SETOMVS FILESYS,FILESYSTEM='NIGELR2.TEST.HFS',SYSNAME=SC64
```

Figure 27. Example of SETOMVS system command

3.3.5 confighfs shell command

The USS confighfs command introduced in DFSMS/MVS 1.5 gives interactive shell users the ability to invoke vfs_pfsctl HFS functions. The vfs_pfsctl function is used to pass control information to the PFS (Physical File System).

The pfsctl (physical file system control — (BPX1PCT)) is a callable service that forwards a command and argument to a physical file system. The meaning of the command and argument are specific to the physical file system and are defined by the physical file system, such as HFS.

Note

In a shared HFS sysplex (OS/390 2.9 and higher), you need to issue the confighfs command from the owning system. If you issue the confighfs from a client system, you receive a message like:

```
Error issuing PFSCTL: RC=0 ERRNO=129(81) REASON=5B360105: HFS is not mounted.
```

The confighfs command calls the PFS directly. On a client system, the PFS does not know anything about the file system. See 7.3, "HFS sysplex sharing (OS/390 V2R9 and above)" on page 164 for additional information regarding HFS sysplex sharing.

The confighfs command is located at path /usr/lpp/dfsms/bin.

For an HFS you can specify:

```
confighfs [-l] [-v n] [-f n] [-q] [pathname] [-x[n] size pathname]
```

The options are:

`-l` Query HFS buffer storage limits.

`-v n` Set virtual buffer storage max to n (where n is in MB). Requires superuser authority.

`-f n` Set fixed buffer storage min to n (where n is in MB). Requires superuser authority.

`-q` Query your HFS global statistics.

`pathname` Query file system statistics for the file system containing each of the path names specified.

`-x size pathname`

Extend the specified file system, where size is the amount to be extended suffixed by the extend unit of M, T, or C (for megabytes, tracks, or cylinders), and pathname is a full or simple pathname to a file or directory in the file system to extend. This requires superuser

authority. Note that this size overrides the secondary allocation (if any) for the HFS.

`-xn size pathname`

Extend the specified file system to a new volume with the same rules as above. This requires superuser authority.

The following are internal debug options:

`-dn` Prints incoming and outgoing pfsctl buffers (where n is 0, 1, or 2).

`-t` Skips issuing the pfsctl.

3.3.5.1 confighfs output description

Figure 28 is an example of the query HFS limits (confighfs -l).

```
STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs -l
HFS Limits
 Maximum virtual storage: _______393(MB)
 Minimum fixed storage:   _________0(MB)
```

Figure 28. Query HFS limits (confighfs -l)

The data returned for the HFS limits option is:

- **Maximum virtual storage**: Specifies the limit for HFS I/O buffers.
- **Minimum fixed storage**: Specifies the value of minimum fixed storage.

Figure 29 is an example of the query global HFS statistics (confighfs -q).

```
STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs -q
HFS Statistics
( 07/06/99 9:10pm )
 Virtual storage:    _______677(pages)
                    _2.6445313(MB)
 Fixed storage:      _________0(pages)
                     _________0(MB)
 Lookup cache hit:   __________________257
 Lookup cache miss:  __________________103
 1st data page hit:  __________________784
 1st data page miss: __________________90

 Pool  Size #DS   BP_pages      Fixed   Already_fixed  Not_already_fixed
    1 ____1 __1 _______133 _________0 _________0 ______2104
    2 ____4 __1 ________16 _________0 _________0 ________18
    3 ___16 __1 ________80 _________0 _________0 ________18
    4 ___64 __1 _______448 _________0 ________46 ________38
```

Figure 29. Query global HFS statistics (confighfs -q)

The data returned for the HFS statistics option is:

- **Virtual Storage:** Specifies the total amount (in pages and MB) of virtual storage assigned to HFS I/O buffers.
- **Fixed Storage:** Specifies the total amount (in pages and MB) of permanently fixed storage assigned to HFS I/O buffers.

- **Lookup cache hit:** The number of times the metadata for a file was found in virtual storage (cache) during file lookup.
- **Lookup cache miss:** The number of times the metadata for a file was not found in virtual storage (cache) during file lookup so I/O was required.
- **1st data page hit:** The number of times the first page of a data file was requested and found in virtual storage (cache).
- **1st data page miss:** The number of times the first page of a data file was requested and not found in virtual storage (cache) thus I/O was required.
- **Pool Number:** Buffer pool ID. Designates one of the four HFS buffer pools.
- **Size:** Buffer size for this pool (in pages).
- **#DS:** Number of data spaces comprising this buffer pool.
- **BP_pages:** Number of pages in this buffer pool currently in use.
- **Fixed:** Number of permanently fixed pages in this buffer pool.
- **Already_fixed**: Number of times a buffer was already fixed prior to an I/O request in this buffer pool. This is a counter which is never decremented.
- **Not_already_fixed**: Number of times a buffer was not already fixed prior to an I/O request in this buffer pool. This is a counter which is never decremented.

Figure 30 is an example of the query file system statistics (configfhfs pathname).

```
STYRES1 @ SC64:/usr/lpp/dfsms/bin>configfhfs /
Statistics for file system HFS.OS390R7.SC64.O37RB1.ROOT
( 07/06/99 9:05pm )
 File system size:____158040
                  _617.34375(MB)
 Used pages:      ____146706
                  _573.07031(MB)
 Attribute pages: ______1882
                  _7.3515625(MB)
 Cached pages:    _________1
                  0.00390625(MB)
 Seq I/O reqs:    ___________________40
 Random I/O reqs: ___________________0
 Lookup hit:      ___________________196
 Lookup miss:     ___________________71
 1st page hit:    ___________________199
 1st page miss:   ___________________40
 Index new tops:  ___________________0
 Index splits:    ___________________0
 Index joins:     ___________________1
 Index read hit:  _________________1800
 Index read miss: ___________________24
 Index write hit: ___________________194
 Index write miss:___________________0
 RFS flags        ___________________82(HEX)
 RFS error flags: ___________________0(HEX)
 High foramt RFN: _________________23E4F(HEX)
 Member count:    _________________17916
 Sync interval:   ___________________30(seconds)
```

Figure 30. Query file system statistics (configfhfs pathname)

The data returned for the file system statistics option is:

- **File system size**: Amount of storage allocated to this HFS file system (in pages and MB).
- **Used pages**: Amount of storage internally used within HFS for data files, directories and HFS internal structures (in pages and MB).
- **Attribute pages**: Amount of storage used for the attribute directory(AD). This number is included in the *Used pages* field (in pages and MB).

  Note: The attribute directory is the internal HFS structure (index) which contains attribute information about individual file system objects as well as attributes of the file system itself.
- **Cached pages**: Amount of data buffer storage cached by the file system (in pages and MB).
- **Seq I/O reqs**: Number of sequential file data I/O requests which have been issued.

  Note: A sequential I/O is one of a series of I/Os to read or write a data file, where the first I/O started at the first byte of the file and each subsequent I/O was for the next sequential set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.
- **Random I/O reqs**: Number of random file data I/O requests that have been issued.

  Note: A random I/O is an I/O that does not read or write the start of a file, and was not preceded by an I/O that read or wrote the immediately preceding set of bytes. This is not meant to imply that actual disk I/O was required; the data may have resided in cache.
- **Lookup hit**: m Number of times the metadata for a file was found in virtual storage (cache) during file lookup.
- **Lookup miss**: Number of times the Metadata for the file was not found in virtual storage (cache) during file lookup and an index call was necessary which may have resulted in an I/O.
- **1st page hit**: The number of times the first page of a data file was requested and found in virtual storage (cache).
- **1st page miss**: The number of times the first page of a data file was requested and was not found in virtual storage (cache), thus, I/O was required.
- **Index new tops**: The number of times the index expanded to the point where an additional level was necessary.
- **Index splits**: The number of index page splits.
- **Index joins**: The number of index page joins.
- **Index read hit**: The number of index read hits.
- **Index read miss**: The number of index read misses.
- **Index write hit**: The number of index write hits.
- **Index write miss**: The number of index write misses.
- **RFS flags**: HFS internal information.
- **RFS error flags**: HFS internal information.
- **High format RFN**: High formatted relative frame number (HFRFN).

- **Member count**: Number of nodes in the file system.

  Note: Please refer to APAR OW39886 (USS configfs COMMAND SHOWS INCORRECT MEMBER COUNT), if your member count is incorrect (too high).
- **Sync interval**: Sync daemon interval.

3.3.5.2 configfs examples

To set virtual and fixed HFS buffer limits to 128MB and 32MB, respectively:

```
configfs -v 128 -f 32
```

To extend the file system for your current directory by 100 cylinders:

```
configfs -x 100c
```

If you need to get statistics for the file systems containing /tmp (as well as root or the current directory), you would enter the following:

```
configfs / /tmp
```

Please refer to Appendix B, “Sample JCL and output” on page 257 for further samples.

3.3.6 Displaying the SYNC interval settings

You can use the MVS system command D OMVS,F to display the current settings.

```
D OMVS,F
BPXO044I 15.04.56 DISPLAY OMVS 113
OMVS     000F ACTIVE          OMVS=(GU)
TYPENAME   DEVICE ----------STATUS----------- MODE QJOBNAME  QPID
TFS             6 ACTIVE                      RDWR
  NAME=/TMP
  PATH=/tmp
  MOUNT PARM=-s 500
HFS             5 ACTIVE                      RDWR
  NAME=OMVS.STYRES1.HFS3
  PATH=/u/guts
  MOUNT PARM=SYNC(120),NOWRITEPROTECT
HFS             4 ACTIVE                      RDWR
  NAME=OMVS.SC64.VAR
  PATH=/var
HFS             3 ACTIVE                      RDWR
  NAME=OMVS.SC64.USERS
  PATH=/u
HFS             2 ACTIVE                      RDWR
  NAME=OMVS.SC64.ETC
  PATH=/etc
HFS             1 ACTIVE                      RDWR
  NAME=HFS.OS390R7.SC64.O37RA1.ROOT
  PATH=/
  MOUNT PARM=SYNC(30)
```

You can also use the USS configfs command to display the sync interval setting for an individual HFS.

The following screen shows the different sync values returned by configfs, based on the examples in 3.3.1, “BPXPRMxx options” on page 56.

```
STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs /
Statistics for file system HFS.OS390R7.SC64.O37RA1.ROOT
( 06/23/99 3:20pm )
...
Sync interval:   _________________30(seconds)

STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs /etc
Statistics for file system OMVS.SC64.ETC
( 06/23/99 3:21pm )
...
Sync interval:   _________________90(seconds)

STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs /u/guts
Statistics for file system OMVS.STYRES1.HFS3
( 06/23/99 3:27pm )
...
Sync interval:   _________________120(seconds)
```

Notes:

1. We eliminated the lines of the configfs output that are not relevant for the sync.
2. The sync interval of the root file system was set by the ROOT statement in BPXPRMxx.
3. The sync interval of HFS data set OMVS.SC64.ETC was derived from the SYNCDEFAULT parameter of the FILESYSTYPE statement.
4. The sync interval of HFS data set OMVS.STYRES1.HFS3 was derived from the SYNC parameter of the associated MOUNT statement.
5. The sync values are always rounded to the next 30-second interval. You will see the values that you have specified and not the adjusted values.

You can also use the configfs -l command to display the current storage limits of an HFS.

```
STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs -l
HFS Limits
 Maximum virtual storage: _______393(MB)
 Minimum fixed storage:   _________0(MB)
```

3.4 Requirements and restrictions

HFS data sets have the following requirements and restrictions:

- They must be system (SMS) managed, reside on DASD volumes managed by the system, and cataloged.

  Note: Non-SMS managed HFS data sets are now supported in DFSMS/MVS 1.4 and 1.5. However, at the present time, these non-SMS managed HFS data sets must be cataloged at mount time, and must be single volume data sets.

- They cannot be processed by standard open, end-of-volume, or access methods; POSIX system calls must be used instead. For exceptions refer to 3.4.2.6, “Processing HFS files with a sequential access method” on page 75.
- They are supported by standard DADSM create, rename, and scratch.
- They are supported by DFSMShsm for dump/restore and migrate/recall and DFSMSdss is used as the data mover.
- They are not supported by the IEBCOPY or the DFSMSdss COPY function.
- RELEASE of HFS data sets is no longer supported, since the DADSM PARTREL function no longer supports HFS data sets.
- The maximum number of files that are supported in an HFS data set is 2**48. Each directory and each file counts against this number. There is no per directory maximum; everything counts against the total for the whole HFS.
- For DFSMS/MVS 1.5 and OS/390 2.7, the maximum number of concurrent mounts may reach about 8000. For DFSMS/MVS 1.5 with earlier releases of OS/390, the limit is likely to be lower, depending on the number of multi-volume HFS data sets. Roughly 1K of storage is used below the line for each mounted file system in the SWA. This can mean that you run out of virtual storage before the TIOT is filled and see a non-restartable wait state of 07E. You can reduce the chance of this happening by allowing automounts to time out, thereby releasing the storage for reuse.

  With DFSMS/MVS 1.4, the normal TIOT size of 32K gives a maximum of 1635 concurrent mounts. A 64K TIOT allows a maximum of 3270 concurrent mounts.

  The TIOT size is controlled by using the ALLOCxx member in PARMLIB.
- A multi-volume HFS can span up to 59 volumes and has 255 extents with up to 123 extents per volume.

3.4.1 Restrictions for executable files (object modules)

- Although you can use the binder to create executable files in a hierarchical file system data set, you cannot create overlay modules. Also, if the executable file you are creating has the same name as an existing file in the data set, the existing file is replaced, even if it is not executable, and even if you specify that existing files should not be replaced.
- Executable files in an HFS data set cannot have alternate entry points. Any alternate entry points specified are considered true aliases.
- When saving an executable file in an HFS data set, the binder does not take the save or stow exits.
- Checkpoints cannot be taken in steps that are using OS/390 UNIX System Services.

3.4.2 Additional dependencies, issues, restrictions, and requirements

Here are additional requirements concerning software, hardware, maintenance, and processing.

3.4.2.1 Software dependencies

OS/390 2.5, or higher, is required for installations to realize the HFS performance benefits associated with the changes in DFSMS/MVS 1.5.

A new FMID (HDZ11EH) is included on the DFSMS/MVS product tape to install DFSMS/USS — an extension to the OS/390 Unix System Services provided for this new support. A separate FMID is required in this case because it must be applied to the system as part of Wave 2 of the OS/390 installation rather than in Wave 1, which includes only the basic elements. After DFSMS and USS are installed, no additional user action is required, as the function becomes integrated with OS/390 UNIX System Services.

3.4.2.2 Hardware dependencies

There are no hardware dependencies.

3.4.2.3 Maintenance issues

It is very important that you apply all the maintenance listed in APAR II11833, the DFSMS/MVS 1.5 program directory, and in the Preventive Service Planning Bucket (PSP; Upgrade ID 'MVSDFSMS1507)).

Note from Information APAR II11833: The following maintenance (listed in Information APAR II11833) is required for the noted DFSMS/MVS 1.5 components. Severe errors have been encountered without this maintenance. This list should be used in addition to the information in the MVSDFSMS 'bucket', and maintenance noted in the DFSMS 1.5 Program Directory.

3.4.2.4 Coexistence issues

Previous releases of DFSMS/MVS cannot mount HFS data sets that span more than one volume. Toleration PTFs must be installed on lower-level systems that share HFS data with DFSMS/MVS 1.5 systems to ensure the integrity of multi-volume HFS data sets.

Single volume HFS data sets created on a DFSMS/MVS 1.5 system are still accessible on previous releases. However, applying toleration PTFs are still recommended to avoid IMF abends due to the write protection feature.

It is also very important that you have all toleration and all HIPER maintenance installed on any systems that are sharing the HFS data sets. The maintenance is referenced in the DFSMS 1.5 Program Directory and in the Preventive Service Planning Bucket (Upgrade ID: MVSDFSMS150).

3.4.2.5 Processing restrictions

A colony address space is one that OS/390 UNIX starts in order to run certain types of physical file systems. This is indicated by the presence of the ASNAME() parameter on the FILESYSTYPE statement that defines the PFS. It is rare for HFS to get started in a colony address space. You should be aware, however, that with the changes introduced by this support, HFS should not be allowed to run in a colony address space.

3.4.2.6 Processing HFS files with a sequential access method

You can access HFS files by using BSAM, QSAM and VSAM access methods.

For BSAM and QSAM, the HFS file is accessed as if it were a single volume, physical sequential data set residing on DASD. For VSAM, the HFS file is accessed as an ESDS. Since HFS files are not actually stored as physical sequential data sets or ESDSs, some processing restrictions might apply, and certain macros and services might have incompatibilities when HFS files are processed.

You can access an HFS file via BSAM or QSAM (DCB DSORG=PS) or VSAM (simulated as an ESDS) by specifying PATH=pathname on the JCL DD statement, SVC 99, or TSO ALLOCATE command.

The following types of HFS files are supported:

- Regular files
- Character special files (null files only)
- FIFO special files
- Symbolic links

The following types of HFS files are not supported:

- Directories
- External links

Data can reside on a system other than the one the user program is running on without using shared DASD. The other system can be MVS or non-MVS. NFS transports the data.

Since HFS files are not actually stored as physical sequential data sets or ESDSs, the system cannot simulate all the characteristics of a sequential data set. Because of this, there are certain macros and services which have incompatibilities or restrictions when dealing with HFS files. For example, DSCBs and UCBs do not exist for HFS files, therefore, any service which relies on a DSCB or UCB for information may not work with these files.

With traditional MVS data sets, other than VSAM linear data sets, the system maintains record boundaries. That is not true with byte-stream files such as HFS files.

Examples of VSAM restrictions are:

- No CI access (MACRF=CNV)
- Only one string
- Only synchronous requests
- Certain SHOWCB requests return dummy values.

Refer to topic 3.7.11 *Accessing HFS Files Via BSAM and QSAM* in the book *DFSMS/MVS Version 1 Release 5 Using Data Sets,* SC26-4922. Also, see *DFSMS/MVS Macro Instructions for Data Sets,* SC26-4913. Information APAR II09184 provides similar information.

Chapter 4. Allocating and mounting HFS data sets

This chapter shows you how to allocate and mount an HFS data set.

4.1 Allocating an HFS

As with other types of MVS data sets, there are various methods available to allocate an HFS data set. To allocate an HFS data set, you need to specify a *Data Set Name Type* (*DSNTYPE*). *DSNTYPE=HFS* indicates an HFS data set.

Two ways are available to specify the Data Set Name Type:

- **Direct** specification of *Data Set Name Type = HFS* during allocations. These provide a way to directly specify *Data Set Name Type = HFS.* For example:
  - ISPF option 3.2.A Data Set Utility - Allocate new data set.
  - DD statement in a batch job.
- **Indirect** specification by assigning an SMS data class in ACS routines that have specified a *Data Set Name Type = HFS.* Depending on your ACS routines, for example, you can assign a data class by explicitly specifying a DATACLAS parameter, or you can derive the data class from a data set name.

  The indirect method is available for all kinds of allocation, because the data class ACS routine will always be executed.

  Some sample indirect allocation methods are:
  - ISPF option 3.2.A Data Set Utility - Allocate new data set.
  - DD statement in a batch job.
  - TSO ALLOCATE command.
  - IDCAMS ALLOCATE.

The direct and indirect methods are also available for multi-volume HFS data set allocation. Most methods provide a way to directly specify a volume count. Or, you can assign a data class in your ACS routines that also indirectly provides a volume count.

If you are not familiar with the standard MVS allocation methods, you can also use the OS/390 ISPF shell (ISHELL) to allocate a single or multi-volume (only indirectly by data class) HFS data set.

HFS data sets should usually be SMS managed, therefore a valid storage class must be assigned in the storage class ACS routine for SMS managed HFS data set allocations. Otherwise, the HFS data set will be allocated as a non-SMS data set.

Note: Non-SMS managed HFS data sets are now supported. However, at the present time, these non-SMS managed HFS data sets must be cataloged at mount time and may not be multi-volume.

HFS data sets can be allocated by tracks, cylinders, or any other SMS-supported method (AVGREC, blocks, and so on). You can achieve very small performance improvement and storage savings by minimizing the number of extents in your HFS data sets.

See 1.2.2, “Index structure of an HFS data set” on page 4 for further information about the size and the structure of an HFS data set.

4.1.1 Multi-volume allocation considerations

For an SMS managed data set, the system uses the storage class attributes and the storage group to select one or more volumes for the data set. For a new SMS managed data set, you can either specify the name of a storage class on the STORCLAS parameter, or you can allow an ACS routine to assign a storage class to the data set. Note that an ACS routine can override the storage class specified through STORCLAS.

4.1.1.1 Specific volume requests for system-managed DASD data sets

You can specify one or more volume serial numbers on the VOLUME parameter if the storage administrator has specified GUARANTEED_SPACE=YES in the storage class. In this case, SMS uses the volumes you explicitly specify. If it cannot, the allocation fails. See 4.1.2, “Guaranteed space considerations” on page 78 for more information.

4.1.1.2 Non-specific volume requests for system-managed DASD data sets

If you specify GUARANTEED_SPACE=NO in the storage class, then SMS chooses the volumes for allocation, ignoring any VOL=SER statements specified on JCL. Primary space on the first volume is preallocated. NO is the default.

You can omit the VOLUME parameter to make a nonspecific volume request for a new system-managed data set. SMS selects the volume to be used for the data set.

4.1.1.3 Requesting volumes

To allocate a multi-volume data set, you can specify *one* of the following:

- A data class that has a volume count greater than one
- A volume count greater than one on the VOLUME or UNIT parameter
- Two or more volume serial numbers

Note: An HFS can have up to a maximum of 255 extents. However, in practice, the actual maximum number will range from 251 to 255 extents. To extend an HFS data set, DADSM is called to obtain additional extents. DADSM will return the requested secondary space amount within one to five extents. Since there is no way to request that DADSM limit the number of extents returned to a number less than five, HFS must ensure that any call to DADSM will not cause the number of extents in the data set to exceed the 255 limit. Therefore, HFS will call DADSM to extend the data set only if the current number of extents is 250 or below. This is compatible with the logic used by VSAM.

4.1.2 Guaranteed space considerations

You can preallocate space for single volume and multi-volume HFS data sets before the job step runs by specifying a storage class with a GUARANTEED SPACE attribute. SMS fails the request if sufficient space is not available.

For a multi-volume system-managed data set, *primary space is preallocated on each of the volumes*. The first volume becomes the primary volume. All remaining volumes become candidate volumes with preallocated space. When the primary

extent on the last volume becomes full, the system attempts to create secondary extents on the last volume.

Important Information

For HFS data sets with GUARANTEED SPACE=YES, secondary extents are allocated only on the last volume of the multi-volume data sets. All volumes except the last one will have only primary extents.

4.1.2.1 Preallocating space for multi-volume data sets

The following JCL allocates a multi-volume HFS data set on volumes selected by SMS. Storage class GSPACE must have GUARANTEED SPACE=YES. This example allocates 100MB on each of four volumes. When all of the allocated space is used for the data set on all volumes, the secondary space is allocated only on the last volume.

```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* NON-SPECIFIC GUARANTEED SPACE MULTIVOLUME ALLOCATION        */
//*****************************************************************/
//DDNAME1  DD DSN=STYRES1.MULTIVOL.GSPACE.HFS,
//            DISP=(NEW,CATLG),
//            STORCLAS=GSPACE,
//            UNIT=(3390,4),
//            SPACE=(1,(100,50,1)),AVGREC=M,
//            DSNTYPE=HFS
```

The ISPF data set information panel illustrates that four extents were allocated. The current allocation shows 400 MB (100MB on each of four volumes).

```
Data Set Name  . . . : STYRES1.MULTIVOL.GSPACE.HFS

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated megabytes : 400
 Storage class  . . . : GSPACE         Allocated extents . : 4
  Volume serial . . . : MHLV09         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . : HFS
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 5
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 1
  1st extent megabytes: 100
  Secondary megabytes : 50
  Data set name type  : HFS
```

If you press enter on the ISPF data set information panel, then you will see the four selected VOLSERs.

Note: ISPF option 3.4 shows a *'+'* to indicate a data set is multi-volume if the data set was allocated with the ability to span volumes. Option 3.2 shows the data set as multi-volume if data has actually been written to two or more volumes. This is

not consistent. Refer to APAR OW37267; ISPF will be changed in the next release to be consistent.

```
 Data Set Information
C .----------- Volume Information -----------.
  |                                           |
D | Command ===>                              | HFS
  |                                           |
G | All allocated volumes:                    | llocation
  |                              More:     +  | d megabytes : 400
  | Number of volumes allocated:  4           | d extents . : 4
  |                                           | dir. blocks : NOLIMIT
  | MHLV09  MHLV03  MHLV07  MHLV05            |
  |                                           |
  |                                           | tilization
  |                                           | es  . . . . : 5
  | F1=Help      F3=Exit     F12=Cancel       | ed  . . . . : 1
  .-------------------------------------------. f members . : 1
  1st extent megabytes: 100
  Secondary megabytes : 50
  Data set name type  : HFS
```

For HFS data sets, secondary extents are allocated only on the last volume of the multi-volume data sets. All volumes except the last one will have only primary extents. The process is as follows:

1. 100MB is used on the first volume.
2. If more space is needed, 100MB of primary space is used on the second volume.
3. The same process is repeated on each volume.
4. Finally, additional secondary space is allocated on the last volume, if sufficient room is available. In this case, the system will attempt to allocate 500MB of space each time extend is attempted. This will continue until either the extent limit is reached or the volume is full. In this example, you may get up to 123 extents on the final volume and only the primary allocation on the first three volumes.

Note: If you allocate an HFS data set using the same JCL, with the exception of the STORCLAS statement, you do not perform a GUARANTEED SPACE request. In this case, only one extent (100MB) on the first (primary) volume will be allocated, and three candidate volumes will be added to the associated catalog entry. ISPF current allocation shows (just after allocating the HFS data set):

- Allocated megabytes: 100
- Allocated extents: 1

However, without GUARANTEED SPACE, the maximum number of extents is between 251 and 255 (123 extents on the first and second volume and five to nine extents on the third volume. For more information, see 4.1.1, “Multi-volume allocation considerations” on page 78.).

4.1.2.2 Honoring specific volume requests

In addition to preallocating space for multi-volume HFS data sets, you can use the GUARANTEED SPACE attribute to let SMS honor a request for single or multiple volumes explicitly specified by the user. For example, you can allow

users to preallocate a multi-volume HFS data set on specific volumes. SMS insures that all the specified volumes are in the same storage group and also that the storage group containing the volumes is one of the storage groups assigned to the storage group ACS routine. You can also use the GUARANTEED SPACE attribute to allocate space specifically on volumes within the same storage group as shown on the next screens.

The sample shows JCL preallocating space for a multi-volume data set on volumes SBOX13, SBOX14, SBOX15 and SBOX16 if the storage class GSPACE has GUARANTEED SPACE = YES:

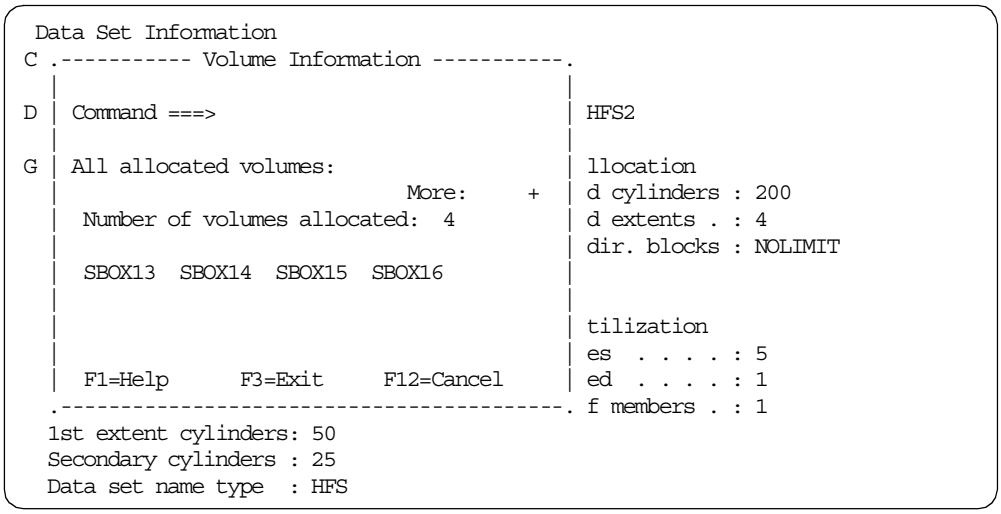
```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* SPECIFIC GUARANTEED SPACE MULTIVOLUME ALLOCATION            */
//*****************************************************************/
//DDNAME1  DD DSN=STYRES1.MULTIVOL.GSPACE.HFS2,
//            DISP=(NEW,CATLG),
//            STORCLAS=GSPACE,
//            VOL=SER=(SBOX13,SBOX14,SBOX15,SBOX16),
//            SPACE=(CYL,(50,25,1)),
//            DSNTYPE=HFS
```

ISPF data set information shows that four extents were allocated. The current allocation is 200 cylinders (50 cylinders on each of the four volumes).

```
Data Set Name  . . . : STYRES1.MULTIVOL.GSPACE.HFS2

General Data                            Current Allocation
 Management class . . : MCDB22           Allocated cylinders : 200
 Storage class  . . . : GSPACE           Allocated extents . : 4
  Volume serial . . . : SBOX13           Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . : HFS
  Organization  . . . : PO              Current Utilization
  Record format . . . : U                Used pages  . . . . : 5
  Record length . . . : 0                % Utilized  . . . . : 1
  Block size  . . . . : 0                Number of members . : 1
  1st extent cylinders: 50
  Secondary cylinders : 25
  Data set name type  : HFS
```

You will see the four VOLSERs when you press enter on the ISPF data set information panel.



Note: For HFS data sets, secondary extents are allocated only on the last volume of the multi-volume data sets. All volumes except the last one will have only primary extents.

If you specify NO for GUARANTEED SPACE, then SMS chooses the volumes for allocation, ignoring any VOL=SER statements specified on JCL. Primary space on the first volume is preallocated. NO is the default for guaranteed space.

End users can allocate space on specific volumes for a data set if you:

- Create at least one storage class with the GUARANTEED SPACE attribute.
- Ensure the user is authorized to the storage class with the GUARANTEED SPACE attribute.
- Write a storage group ACS routine that assigns a storage group containing the volumes explicitly specified by the user (optional).
- Ensure all volumes explicitly specified by the user belong to the same storage group by directing an allocation which is assigned a GUARANTEED SPACE storage class to all the storage groups in the installation.
- Ensure the requested space is available since there is no capability in SMS to allow specific volume requests except with the GUARANTEED SPACE attribute.
- Ensure availability and accessibility specifications in the storage class can be met by the specified volumes.

4.1.3 SMS definition examples

The following examples are based on the SMS constructs defined and explained in 3.1.2, “Defining SMS constructs for HFS data sets” on page 35.

We summarize the SMS definitions used in the later examples:

- DATACLAS=HFS defines a single volume HFS data set (DSNTYPE=HFS).
- DATACLAS=HFSMULT defines a multi-volume HFS data set (DSNTYPE=HFS and VOLCOUNT=3)

- STORCLAS=OPENMVS with minimum definitions to spread allocations across volumes (no Performance Objective (no MSR values); ACCESSIBILITY=NOPREF; ACCESSIBILITY=NOPREF; GUARANTEED SPACE=NO).
- STORCLAS=GSPACE same as STORCLAS=OPENMVS with the exception that GUARANTEED SPACE allocation is allowed (GUARANTEED SPACE=YES).
- MGTMCLAS=HFS with PARTREL=NO ; AUTO BACKUP=YES and BACKUP COPY TECHNIQUE=STANDARD)
- STORAGE GROUP=OPENMVS with AUTO MIGRATE=YES; AUTO BACKUP=YES and AUTO DUMP=NO.
- For indirect allocations:

  The data class ACS routine assigns the HFS data class (with DSNTYPE=HFS) based on the following rules:

  - Pre-assigned DC of HFS (single volume).
  - DSNTYPE=HFS
  - Low level qualifier (LLQ) starting with HFS
  - Pre-assigned DC of HFSMULTI (multivolume) assigns DC HFSMULTI.

  The storage class ACS routine assigns the OPENMVS storage class if:

  - The HLQ of DSNAME is OMVS or HFS.
  - DC HFS or HFSMULTI is defined.

4.1.4 Using ISPF

You can use the ISPF Data Set Utility (option 3.2) to allocate single volume or multi-volume HFS data sets. Depending on your ACS routines, you can use the direct or indirect method to specify DSNTYPE=HFS.

4.1.4.1 Single volume (using ISPF)

The following screen (Sample 1) shows an example of how to directly allocate an HFS data set:

1.) Select option A or M to allocate a new HFS data set.

```
 Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                              Data Set Utility
Option ===> a

    A Allocate new data set               C Catalog data set
    R Rename entire data set              U Uncatalog data set
    D Delete entire data set              S Data set information (short)
blank Data set information                 M Allocate new data set
                                           V VSAM Utilities

ISPF Library:
   Project  . .
   Group  . . .
   Type . . . .

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . . 'Stryes1.user.hfs.data.set
   Volume Serial . . .          (If not cataloged, required for option "C")
```

Figure 31. Sample 1

2.) Enter your space values and specify DSNTYPE=HFS.

```
 Allocate New Data Set
 Command ===>
                                                                More:     +
 Data Set Name  . . . : STRYES1.USER.HFS.DATA.SET

 Management class . . .                (Blank for default management class)
 Storage class  . . . .                (Blank for default storage class)
  Volume serial . . . .                (Blank for system default volume) **
  Device type . . . . .                (Generic unit or device address) **
 Data class . . . . . .                (Blank for default data class)
  Space units . . . . . cyls           (BLKS, TRKS, CYLS, KB, MB, BYTES
                                        or RECORDS)
  Average record unit                  (M, K, or U)
  Primary quantity  . . 5              (In above units)
  Secondary quantity    5              (In above units)
  Directory blocks  . . 0              (Zero for sequential data set) *
  Record format . . . . U
  Record length . . . . 0
  Block size  . . . . . 0
  Data set name type  : HFS            (LIBRARY, HFS, PDS, or blank)  *
```

After you have pressed enter to allocate the HFS data set, you can just press enter (blank - data set information) again to verify your new HFS data set allocation. If allocation was successful, the Data Set Name Type field will show HFS. Note that minimally, an SC must be assigned to this HFS data set for it to be system-managed.

```
Data Set Information
Command ===>

Data Set Name  . . . : STRYES1.USER.HFS.DATA.SET

General Data                          Current Allocation
 Management class . . : HFS            Allocated cylinders : 5
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX15         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . : HFS
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 5
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 1
  1st extent cylinders: 5
  Secondary cylinders : 5
  Data set name type  : HFS
```

Note: If your allocated space differs from the amount of space that you have requested, first check the Default Device Geometry of your SMS installation.

The Default Device Geometry attribute converts an allocation request from tracks or cylinders to KBs or MBs when an esoteric unit is used, or when no unit is given. Through this conversion, uniform space can be allocated on any device type for a given allocation.

Two variations of UNIT coding are:

- Users specify a generic name, such as 3380 or 3390:

  These users have allocation converted to bytes, based on the geometry of that device.

- Users specify an esoteric name, such as SYSDA:

  These users have allocation converted to bytes, based on the Default Device Geometry attribute.

Refer to *DFSMS/MVS V1R5 DFSMSdfp Storage Administration. Reference,* SC26-4920, for more information regarding Default Device Geometry.

4.1.4.2 Multi-volume (using ISPF)

To allocate a new multi-volume HFS data set, you can use the indirect method by selecting an appropriate data class including DSNTYPE=HFS and a volume count greater than one.

Note: Currently, ISPF does not support the direct allocation of HFS multi-volume data sets as is available for other kinds of MVS data sets. You will receive a message `"Invalid combination"`, if you specify a slash (`"/" Allocate Multiple Volumes`) at the ISPF option 3.2 allocation panel to enter more than one volume for an HFS data set.

The following example (Sample 2) uses a pre-assigned data class, HFSMULTI, to allocate a multi-volume HFS data set.

```
 Allocate New Data Set
Command ===>
                                                                More:
Data Set Name  . . . : STRYES1.MULTIVOL.HFS.DATA.SET

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . .                (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . . HFSMULTI       (Blank for default data class)
 Space units . . . . . MB             (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 8              (In above units)
 Secondary quantity    4              (In above units)
 Directory blocks  . . 1              (Zero for sequential data set) *
 Record format . . . . U
 Record length . . . . 0
 Block size  . . . . . 0
 Data set name type  :                (LIBRARY, HFS, PDS, or blank)  *
```

Figure 32. Sample 2

Note: If you do not directly specify a DSNTYPE = HFS, then you need to specify at least one directory block. The number of directory blocks must be specified to indicate that this is a partitioned organized data set, but the value has no effect on allocation. Otherwise, a sequential data set will be allocated.

4.1.5 Using DD statement in a batch job

The DD (data definition) statement in a batch job also provides the ability to directly allocate a single or multi-volume HFS data set. Nevertheless, you can also indirectly allocate an HFS data set.

See 3.1.7, “ACS routines” on page 46 for more examples.

4.1.5.1 Single volume (using DD)

The next example screen shows how to allocate a single volume HFS data set directly, by assigning a DSNTYPE=HFS (Sample 3); and indirectly, by deriving the DC HFS from the LLQ =HFS* (Sample 4).

```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* SAMPLE #3 SINGLE VOLUME BY USING DSNTYPE=HFS                  */
//*****************************************************************/
//DDNAME1  DD DSN=STYRES1.HFS.DATA.SET3,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(1,(2,2,1)),AVGREC=M,
//            DSNTYPE=HFS
//*****************************************************************/
//* SAMPLE #4 SINGLE VOLUME DUE TO LLQ OF HFS*                    */
//*****************************************************************/
//DDNAME2  DD DSN=STYRES1.USER.HFS4,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(TRK,(2,2,1))
```

Figure 33. Sample 3 and sample 4

Note: The number of directory blocks must be specified to allocate an HFS data set, although the value has no real effect on the allocation.

4.1.5.2 Multi-volume (using DD)

In the following example screen, Sample 5 shows how to allocate (indirectly) a multi-volume HFS data set by specifying DSNTYPE=HFS, and a volume count in the UNIT statement. Sample 6 uses the indirect method to allocate a multi-volume HFS data set by specifying DATACLAS=HFSMULTI.

```
//ALLOC1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//*****************************************************************/
//* SAMPLE #5 MULTIVOLUME BY USING VOLCOUNT ON UNIT STATEMENT   */
//*****************************************************************/
//DDNAME3  DD DSN=STYRES1.MULTIVOL.HFS.DATA.SET5,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA,2),
//            SPACE=(CYL,(2,2,1)),
//            DSNTYPE=HFS
//*****************************************************************/
//* SAMPLE #6 MULTIVOLUME BY USING DATACLAS=HFSMULTI              */
//*****************************************************************/
//DDNAME4  DD DSN=STYRES1.MULTIVOL.HFS.DATA.SET6,
//            DISP=(NEW,CATLG),
//            UNIT=(SYSDA),
//            SPACE=(TRK,(4,2,1)),
//            DATACLAS=HFSMULTI
```

Figure 34. Sample 5 and sample 6

Note: The number of directory blocks must be specified to allocate an HFS data set, although the value has no real effect on the allocation.

4.1.6 Miscellaneous allocation methods

As mentioned before, you can use most allocation methods to indirectly allocate a single or multi-volume HFS data set indirectly.

The next screen provides a few examples for the TSO ALLOCATE command and IDCAMS ALLOCATE.

4.1.6.1 Single volume (using other methods)

IDCAMS ALLOCATE (Sample 7): The HFS data class will be derived from the LLQ of the data set name.

```
//STEP  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//*********************************************************************/
//*   SAMPLE #7 INDIRECT ALLOCATION DUE TO LLQ OF HFS*                 */
//*********************************************************************/
//SYSIN    DD *
 ALLOC -
       DSNAME('STYRES1.USER.HFS7') -
       NEW CATALOG -
       SPACE(5,5) CYLINDERS -
       DSORG(PO)
```

Figure 35. Sample 7

The syntax of the TSO ALLOCATE command (Sample 8) is similar to the IDCAMS ALLOCATE. In this example, we have specified DATACLAS(HFS) to force an HFS data set allocation.

```
//STEP1 EXEC PGM=IKJEFT01
//SYSPRINT  DD  SYSOUT=*
//SYSTSPRT  DD  SYSOUT=*
//*********************************************************************/
//*   SAMPLE #8 INDIRECT ALLOCATION BY SPECIFYING DATACLAS(HFS)        */
//*********************************************************************/
//SYSTSIN  DD *
 ALLOC -
       DSNAME('STYRES1.HFS.DATA.SET8') -
       NEW -
       SPACE(3,2) AVBLOCK(1) AVGREC(M) -
       DSORG(PO) -
       DATACLAS(HFS)
/*
```

Figure 36. Sample 8

4.1.6.2 Multi-volume (using other methods)

Sample 9 shows how to allocate an HFS data set by using IDCAMS ALLOC and specifying a DC HFSMULTI. Instead of specifying DATACLAS(HFSMULTI), you can also specify parameter UCOUNT(2) to allocate a multi-volume HFS data set.

```
//STEP  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//*********************************************************************/
//*   SAMPLE #9     MULTIVOLUME HFS   BY SPECIFYING  DATACLAS=HFSMULTI    */
//*********************************************************************/
//SYSIN    DD *
 ALLOC -
       DSNAME('STYRES1.MULTIVOL.HFS.DATA.SET9') -
       NEW CATALOG -
       SPACE(20,10) TRACKS -
       DSORG(PO) -
       DATACLAS(HFSMULTI)
/*
```

Figure 37. Sample 9

Sample 10 shows a multi-volume allocation for a TSO ALLOCATE command by explicitly specifying a volume count (UCOUNT(4)).

```
//STEP1 EXEC PGM=IKJEFT01
//SYSPRINT  DD  SYSOUT=*
//SYSTSPRT  DD  SYSOUT=*
//*********************************************************************/
//*   SAMPLE #10  MULTIVOLUME DUE TO LLQ HFS* & PARAMETER UCOUNT(4)       */
//*********************************************************************/
//SYSTSIN  DD *
 ALLOC -
       DSNAME('STYRES1.USER.MULTIVOL.HFS10') -
       NEW -
       SPACE(5,5) CYLINDERS -
       UCOUNT(4) -
       DSORG(PO)
/*
```

Figure 38. Sample 10

4.1.7 Using ISHELL

The OS/390 ISPF shell (ISHELL) also provides the ability to allocate a single volume or multi-volume HFS data set. However, only the indirect method may be used for a multi-volume HFS.

4.1.7.1 Single volume (using ISHELL)

Using the file system pulldown menu, you can allocate a new HFS data set.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
----------------------------------- .----------------------. ----------
                             OpenMVS I | 2  1. Mount table...  |
                                       |    2. New...         |
Enter a pathname and do one of these:  |    3. Mount(O)...    |
                                       .----------------------.
    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                              More:     +
   /
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________
```

The name of a file system should be a fully qualified name of a data set. Enter the name of a data set, and the number of cylinders for primary and secondary allocation.

Sample 11 allocates an HFS data set with only a primary allocation. The HFS will not automatically be extended if no secondary allocation is specified. This limits the size of an HFS to the primary allocation amount.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
- .-------------------------------------------------------------.
  |                          Make a File System                 |
  |                                                             |
E |  File system name  . . . . 'STYRES1.USER.HFS.DATA.SET11'    |
  |  Primary cylinders . . . . 5_______                         |
  |  Secondary cylinders . . . 0_______                         |
  |  Storage class . . . . . . ________                         |
  |  Management class  . . . . ________                         |
  |  Data class  . . . . . . . ________                         |
R |                                                             |
  |   F1=Help       F3=Exit       F6=Keyshelp  F12=Cancel       |
  .-------------------------------------------------------------.
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________
```

Figure 39. Sample 11

Note: ISHELL specifies DSNTYPE=HFS during the allocation process. but a valid storage class must be specified either directly or by the storage class ACS routine if you want the HFS data set to be system-managed.

4.1.7.2 Multi-volume (using ISHELL)

To allocate a multi-volume HFS data set using ISHELL, you can assign a DC which has already been defined with a volume count > 1. In Sample 12, the pre-assigned DC HFSMULTI allocates a multi-volume HFS data set on three volumes (2 candidate volumes).

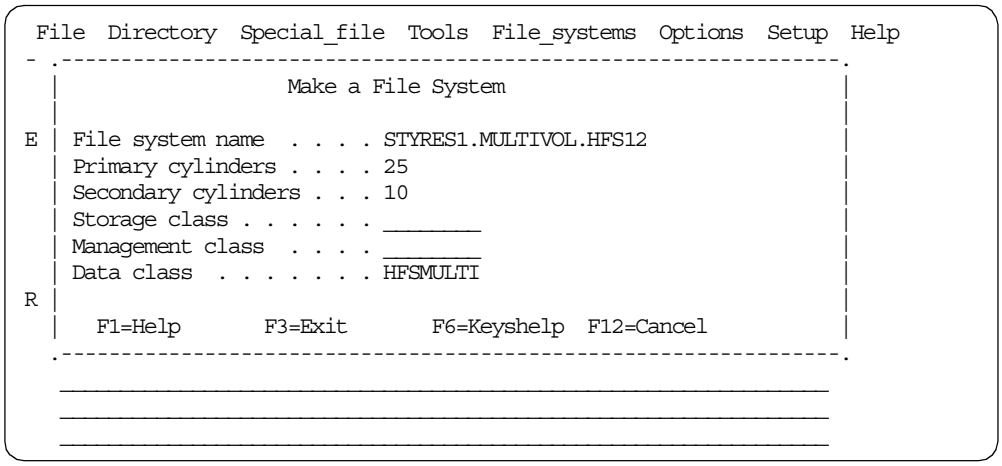


Figure 40. Sample 12

4.1.8 Summary of HFS data set allocations

Table 5 summarizes the assigned SMS classes and HFS data set attributes after allocation.

Table 5. Summary of sample HFS allocations

Sample	Data class	Type of allocation	Primary amount	Secondary amount	Single or multi-volume	Volume count
#1	HFS	cylinder	5	5	single	1
#2	HFSMULTI	MB	8	4	multi	3
#3	HFS	MB	2	2	single	1
#4	HFS	track	2	2	single	1
#5	HFS	cylinder	2	2	multi	2
#6	HFSMULTI	track	4	2	multi	3
#7	HFS	cylinder	5	5	single	1
#8	HFS	MB	3	2	single	1
#9	HFSMULTI	track	20	10	multi	3
#10	HFS	cylinder	5	5	multi	4
#11	HFS	cylinder	5	0	single	1
#12	HFSMULTI	cylinder	25	10	multi	3

4.2 Mounting an HFS

After an HFS data set is allocated, you must mount it at a mount point that is off the root directory in order to make it available.

An HFS data set is a mountable file system. A mountable file system can be logically mounted to a mount point, which is a directory in another file system, with a TSO MOUNT command. A mountable file system can be unmounted from

a mount point with a TSO UNMOUNT command. ISPF user and OMVS shell users can also perform these tasks using their unique methods.

This topic provides information about what should be done before and after mounting an HFS and how to mount an HFS. Furthermore, we describe how to determine the status of mounted HFS data sets.

The restrictions for mounting file systems are:

- The mount point must be a directory. If it is not an empty directory, files in that directory are not accessible while the file system is mounted.
- Only one file system can be mounted at a directory (mount point) at any one time.
- If the file system is to be shared by multiple systems, refer to Chapter 7, “Sharing and serialization for HFS data sets” on page 167.

You can customize the *automount facility* to mount all user file systems automatically when they are needed. This is the preferred method to manage user HFS data sets because it saves administration time. For more information, refer to A.2, “Automount facility” on page 248.

Important Information

A mount and unmount user must have superuser authority, or at least have *READ* access to the *BPX.SUPERUSER FACILITY* class.

4.2.1 Before mounting HFS data sets

You must build a directory in the root file system, or elsewhere in the hierarchy, before mounting an HFS data set. A directory is used as a mount point for a mountable file system. The mount point should be an empty directory; Otherwise, its contents will be hidden (not deleted) for the duration of any subsequent mounts. To build the directory, use one of the following:

- *TSO MKDIR* command interactively; in an in-stream data set in the JCL, such as SYSIN; or in a CLIST or REXX exec
- A *mkdir* OMVS shell command
- The *ISHELL* directory pull-down

Figure 41 shows how to make a new directory /u/styres2, by using the TSO MKDIR command:

```
                              ISPF Command Shell
Enter TSO or Workstation commands below:

===> MKDIR '/u/styres2'



Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
=>
=>
=>
```

Figure 41. TSO MKDIR command example

The permission bits for directory /u/styres2 are set to 755. For more information about default permissions set by the system, refer to 2.4.3.2, “Default permissions set by OS/390” on page 28.

4.2.2 TSO MOUNT and UNMOUNT commands

You can use the TSO MOUNT command under a superuser ID to mount the new file system over the directory of an existing file system.

The MOUNT command format is as follows:

```
MOUNT FILESYSTEM(file_system_name)
            MOUNTPOINT(pathname)
            TYPE(file_system_type)
            MODE(RDWR|READ)
            PARM(parameter_string)
            SETUID|NOSETUID
            WAIT|NOWAIT
            SECURITY|NOSECURITY
```

See 3.3.2, “TSO MOUNT command” on page 63 for a description of the TSO MOUNT parameters, including the two new PARM keywords:

```
PARM('SYNC(t),NOWRITEPROTECT')
```

The UNMOUNT command removes a file system from the file system hierarchy. The alias for this command is UMOUNT. The UNMOUNT command format is as follows:

```
UNMOUNT FILESYSTEM(file_system_name)
          DRAIN | FORCE | IMMEDIATE | NORMAL |
          REMOUNT(RDWR | READ) | RESET
```

See 3.3.3, “TSO UNMOUNT command” on page 64 for a description of the TSO UNMOUNT command.

For example, the directory /u/styres2 is a mount point for OMVS.SC64.STYRES2 with read/write mode and sync interval of 120 seconds. Figure 42 and Figure 43 show examples of TSO MOUNT and UNMOUNT commands.

```
                                ISPF Command Shell
 Enter TSO or Workstation commands below:

 ===> MOUNT FILESYSTEM('OMVS.SC64.STYRES2') TYPE(HFS)
MOUNTPOINT('/u/styres2') PARM('SYNC(120)')


 Place cursor on choice and press enter to Retrieve command

=>
=>
=>
=>
=>
=>
=>
```

Figure 42. TSO MOUNT command example

```
                                ISPF Command Shell
 Enter TSO or Workstation commands below:

 ===> UNMOUNT FILESYSTEM('OMVS.SC64.STYRES2')


 Place cursor on choice and press enter to Retrieve command

=>
=>
=>
 =>
=>
=>
=>
```

Figure 43. TSO UNMOUNT command example

Important Information

With OS/390 2.8, there is a new UNIXPRIV class which contains profiles that allow users to mount (for example) without being a superuser. The profile name is SUPERUSER.FILESYS.MOUNT.

Prior to OS/390 2.7, TSO MOUNT and UNMOUNT commands can only be performed with superuser (UID=0) authority or after changing the effective UID by use of the ISHELL.

For more information about how to set an effective UID in the ISHELL, please refer to 4.2.3, “ISPF ISHELL” on page 95.

4.2.3 ISPF ISHELL

You can invoke the ISPF ISHELL by typing the TSO ISHELL command. This is a panel interface for performing many user and administrator tasks with USS user IDs and the HFS file system. If you are more comfortable using the ISPF editor and ISPF pull-down menus, the ISHELL is the tool for you.

Note: You must have an OMVS segment in your RACF profile to use the ISHELL.

If you are not a superuser, but have permission for read access to the facility class BPX.SUPERUSER, you have to set effective UID (SETEUID) to 0 (zero) before mounting HFS.

The next example shows how to set effective UID to 0 (zero) in the ISHELL.

From ISPF option 6, type ISHELL, bring the cursor up to the Setup pull-down menu item and press Enter.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
-------------------------------------------------------------------------------
                            OpenMVS ISPF Shell

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                                    More:     +
   /u/styres2
   ___________________________________________________________________________
   ___________________________________________________________________________
   ___________________________________________________________________________



(C) Copyright IBM Corp., 1993. All rights reserved.
Command ===> ___________________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
```

Select option 7, `Enable superuser mode`, and then press Enter.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
---------------------------------------- .---------------------------------.
                           OpenMVS ISPF S | 7  1. *User...                  |
                                          |    2. *User list...             |
Enter a pathname and do one of these:     |    3. *All users...             |
                                          |    4. *All groups...            |
    - Press Enter.                        |    5. *Permit field access...   |
    - Select an action bar choice.        |    6. *Character Special...     |
    - Specify an action code or command on |    7. Enable superuser mode(SU) |
                                          .---------------------------------.
Return to this panel to work with a differ .---------------------------------.
                                          | Some choices (*) require        |
   /u/styres2                             | superuser or the "special"      |
   ______________________________________ | attribute for full function, or |
   ______________________________________ | both                            |
   ______________________________________ .---------------------------------.




Command ===> ____________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
```

You are now set as effectively having UID 0 (zero) and have the authority to mount and unmount HFS file systems.

The next example shows how to mount and unmount HFS file systems.

In the ISHELL menu screen, bring the cursor up to the File_system directory pull-down menu item and press Enter.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
------------------------------------------------------------------------------
                           OpenMVS ISPF Shell

Enter a pathname and do one of these:

    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                                More:     +
   /u/styres2
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________




(C) Copyright IBM Corp., 1993. All rights reserved.
Command ===> ____________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
```

Select option 3, `Mount(0)`, and then press Enter.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
----------------------------------- .----------------------. ----------
                             OpenMVS I | 3  1. Mount table...  |
                                       |    2. New...         |
Enter a pathname and do one of these:  |    3. Mount(O)...    |
                                       .----------------------.
    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                                  More:     +
   /u/styres2
   _____________________________________________________________________
   _____________________________________________________________________
   _____________________________________________________________________




Command ===> _____________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
```

Insert the file system name, type, and new owner, and then press Enter.

```
 File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
- .-------------------------------------------------------------------------.
  |                          Mount a File System                            |
  |                                                                         |
E | Mount point:                                                            |
  |                                                        More:     +      |
  |    /u/styres2                                                           |
  |    _____________________________________________________________       |
  |    _____________________________________________________________       |
  |    _____________________________________________________________       |
R |                                                                         |
  | File system name . . OMVS.SC64.STYRES2_____________________________     |
  | File system type . . HFS_____                                           |
  | New owner  . . . . . STYRES2_                                           |
  |                                                                         |
  | Select additional mount options:                                        |
  | _  Read-only file system                                                |
  | _  Ignore SETUID and SETGID                                             |
  | _  Bypass security                                                      |
  | Mount parameter:                                                        |
  | SYNC(120)_____________________________________________________          |
C |  F1=Help       F3=Exit       F4=Name       F6=Keyshelp  F12=Cancel      |
  .-------------------------------------------------------------------------.
F10=Actions  F11=Command  F12=Cancel
```

The data set, OMVS.SC64.STYRES2 is now mounted on /u/styres2, and the sync interval is set to 120 seconds.

The owner of mount point directory of /u/styres2 is now automatically set to STYRES2 by ISHELL's *new owner* parameter.

4.2.4 OMVS mountx and unmountx shell commands

In the shell, the MOUNT and UNMOUNT commands are available as TSO commands, not shell commands. However, you can also issue those commands from the shell, using the TSO shell command. Additionally, the mountx and unmountx REXX programs that can be run from the shell are in the directory /samples.

In the shell, if you are not superuser but have read access to BPX.SUPERUSER, you have to use the su command to switch your authority before any mount or unmount operations.

Type OMVS from ISPF option 6 to enter the shell and execute the commands to mount and unmount the HFS data set OMVS.SC64.STYRES2.

Figure 44 shows an example of the commands required, including issuing the /samples/mountx and /samples/unmountx REXX exec from the shell.

```
$ su
# /samples/mountx -p'SYNC(120)' /u/styres2 omvs.sc64.styres2
OMVS.SC64.STYRES2 is now mounted at
/u/styres2
# /samples/unmountx /u/styres2
Unmount complete for OMVS.SC64.STYRES2
#
 ===>
```

Figure 44. mountx and unmountx shell command examples

4.2.5 After mounting HFS data sets

After mounting the new HFS data set, you should enter the shell and set the file permissions and owners for each file's directory. The chmod command can be used to set these permissions. The chown command can be used to set the new owner.

When you mount a new HFS data set for the first time, the permission bits of the mounted directory are overridden from their original (usually 755) bits to 700. Therefore, you should change the permission bits after mounting the new HFS.

Unless you mount the new HFS by using ISHELL's new owner parameter, you should change the owner name after mounting the new HFS.

Figure 45 shows an example of how to change the file permissions, owner, and group name for the /u/styres2 directory.

```
STYRES2 @ SC64:/u>ls -al
total 80
drwx------   4 HAIMO    SYS1        8192 Jun 17 13:16 .
drwxr-xr-x  13 HAIMO    TTY         8192 Mar  5 15:51 ..
drwx------   2 HAIMO    SYS1        8192 Jun 10 18:54 styres2
drwxrwxrwx   2 HAIMO    SYS1        8192 Jun 17 13:16 styres3
STYRES2 @ SC64:/u>chmod 755 /u/styres2
STYRES2 @ SC64:/u>chown styres2:itsosj /u/styres2
STYRES2 @ SC64:/u>ls -al
total 80
drwx------   4 HAIMO    SYS1        8192 Jun 17 13:16 .
drwxr-xr-x  13 HAIMO    TTY         8192 Mar  5 15:51 ..
drwxr-xr-x   2 STYRES2  ITSOSJ      8192 Jun 10 18:54 styres2
drwxrwxrwx   2 HAIMO    SYS1        8192 Jun 17 13:16 styres3
STYRES2 @ SC64:/u>
```

Figure 45. chmod and chown shell command examples

You can also use ISHELL to change these attributes.

If you change these attributes once, you do not need to change them when the HFS is mounted again later.

4.2.6 Adding MOUNT Statements to BPXPRMxx

To have mountable file systems logically mounted when the system IPLs, add MOUNT statements to the BPXPRMxx PARMLIB member.

If you want to make the mounting of the OMVS.SC64.STYRES2 data set permanent, you must add an entry in the BPXPRMxx member of SYS1.PARMLIB. These mount statements should follow the ROOT statement for the root file system.

```
MOUNT FILESYSTEM('OMVS.SC64.STYRES2')
      TYPE(HFS)
      MOUNTPOINT('/u/styres2')
      MODE(RDWR)
       PARM('SYNC(120)')
```

4.2.7 Status of mounted HFS data sets

When the mount operation is done, you can determine if the mount has completed with one of the following commands:

- The df shell command
- The DISPLAY OMVS,F operator command
- The *MOUNT table option* on the File Systems pulldown in the ISPF Shell (accessed by the ISHELL command)

Examples of these commands are shown in Figure 46, Figure 47, and Figure 48.

```
STYRES2 @ SC64:/>df
Mounted on     Filesystem                Avail/Total    Files      Status
/tmp           (/TMP)                    999797/1000000 127974     Available
/u/styres2     (OMVS.SC64.STYRES2)       7160/7200      4294967294 Available
/var           (OMVS.SC64.VAR)           12848/12960    4294967290 Available
/u             (OMVS.SC64.USERS)         37368/37440    4294967292 Available
/etc           (OMVS.SC64.ETC)           74072/76320    4294967058 Available
/              (HFS.OS390R7.SC64.O37RA1.ROOT)89016/1264320  4294949381 Available
STYRES2 @ SC64:/>
 ===>
```

Figure 46. df shell command

```
D OMVS,F
BPXO044I 04.27.14 DISPLAY OMVS 693
OMVS     000F ACTIVE          OMVS=(7B)
TYPENAME   DEVICE ----------STATUS----------- MODE QJOBNAME  QPID
TFS             5 ACTIVE                      RDWR
  NAME=/TMP
  PATH=/tmp
  MOUNT PARM=-s 500
HFS            10 ACTIVE                      RDWR
  NAME=OMVS.SC64.STYRES2
  PATH=/u/styres2
  MOUNT PARM=SYNC(120)
HFS             4 ACTIVE                      RDWR
  NAME=OMVS.SC64.VAR
  PATH=/var
HFS             3 ACTIVE                      RDWR
  NAME=OMVS.SC64.USERS
  PATH=/u
HFS             2 ACTIVE                      RDWR
  NAME=OMVS.SC64.ETC
  PATH=/etc
HFS             1 ACTIVE                      RDWR
  NAME=HFS.OS390R7.SC64.O37RA1.ROOT
  PATH=/
```

Figure 47. DISPLAY OMVS,F operator command

```
 Work with Mounted File Systems

Select one or more file systems with / or action codes.
  U=Unmount   A=Attributes   C=Change mode   R=Reset unmount or quiesce
  File system name                              Status        Row 1 of 7
_ /TMP                                          Available
_ HFS.OS390R7.SC64.O37RA1.ROOT                  Available
_ OMVS.SC64.ETC                                 Available
_ OMVS.SC64.STYRES2                             Available
_ OMVS.SC64.USERS                               Available
_ OMVS.SC64.VAR                                 Available









Command ===> ____________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F11=Command  F12=Cancel
```

Figure 48. ISHELL mount table option

Chapter 5. Managing HFS data sets

This chapter provides information about managing HFS data sets. We include examples of:

- Backup and restore processing of an HFS data set that is mounted read/write
- Recovery processing
- Increasing the file size of an HFS data set
- Converting from single to multi-volume HFS data sets
- Interpreting the different kinds of file size displays

Backup and restore processing for individual files in an HFS by using Tivoli Storage Manager is also discussed in Chapter 6, “Tivoli Storage Manager” on page 153. Usage of pax, tar, and cpio USS commands is discussed in A.1, “The pax, tar and cpio commands” on page 245.

For both SMS managed and non-SMS managed HFS data sets, DFSMShsm can perform automatic volume backup (by invoking DFSMSdss) and incremental backup.

Note: The DFSMSdss COPY function is not supported for HFS data sets.

5.1 DFSMSdss dump and restore

To back up and restore an entire HFS data set (not individual files within the HFS data set), you can use DFSMSdss functions:

1. Use the DFSMSdss DUMP function to dump the file system. The logical dump processing is the recommended method to dump an HFS data set.
2. Use the DFSMSdss RESTORE function to restore the dumped file system with a new name (rename). If you want to maintain the original file system name, you must first unmount the HFS. You can then replace the original HFS data set during RESTORE.

You can use the MVS D OMVS,F system command to display the currently mounted HFS data sets.

For additional information regarding DFSMSdss processing and control statements, refer to:

- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Reference,* SC26-4929
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Guide,* SC26-4930

5.1.1 Dump processing

DFSMSdss provides four different kinds of backup processing:

- Logical data set dumps
- Physical data set dumps
- Logical volume dumps
- Physical volume dumps

DFSMSdss can perform either logical or physical processing. If you dump a data set logically, DFSMSdss restores it logically; if you dump it physically, DFSMSdss restores it physically.

Logical processing operates against data sets independently of physical device format. The data sets are located by searching either the catalog or VTOC. If input volumes are specified through the LOGINDD, LOGINDYNAM, or STORGRP keywords, then data sets are located by searching the VTOC(s). Otherwise, data sets are located by searching the catalog.

Physical processing moves data at the track-image level and operates against volumes, tracks, and data sets. The data sets are located by searching the VTOC.

The processing method is determined by the keywords specified on the command. For example, LOGINDYNAM indicates a logical dump and INDYNAM a physical dump. Each type of processing offers different capabilities and advantages.

You can select data sets for DFSMSdss processing by filtering on specified criteria. DFSMSdss can filter on fully qualified or partially qualified data set names (by using the INCLUDE or EXCLUDE keyword) and on various data set characteristics (by using the BY keyword; for example `BY(DSORG,EQ,HFS)`).

You can filter data sets with any of the following commands:

- Logical dump
- Logical restore
- Physical data set dump
- Physical data set restore

With DFSMSdss V1 R5, the new STORGRP keyword for logical data set dump operations allows filtering by storage group name, in addition to filtering by volume serial numbers. It allows you to dump a complete storage group.

5.1.1.1 Logical dump

DFSMSdss performs logical processing if you specify the DATASET keyword with the DUMP command, and either no input volume is specified, or LOGINDDNAME, LOGINDYNAM, or STORGRP is used to specify input volumes.

If you specify the DATASET keyword with the DUMP command and do not specify input volumes (neither LOGINDYNAM nor LOGINDDNAME), DFSMSdss performs a logical data set dump using information in the **catalogs** to select data sets.

If you specify the DATASET keyword with either LOGINDDNAME or LOGINDYNAM, DFSMSdss performs a logical data set dump using information in the **VTOCs** to select data sets.

Important information

Mounted HFS data sets should be backed up using **logical data set dump**.

You can dump an HFS data set from any system in participating group in a shared HFS sysplex (OS/390 2.9 and higher).

In a pre-2.9 or non-shared environment, you must perform the dump on the *same system* on which the HFS is currently mounted read/write.

Logical data set dump provides the ***quiesce*** serialization mechanism (using the BPX1QSE callable service) to ensure data integrity. The quiesce allows an HFS data set to be dumped while in use. In a non-shared environment, the dump job must be executed on the same system on which the HFS is currently mounted.

Part of the quiesce processing is to perform a ***sync***. This is intended to flush out any buffered data to DASD prior to a logical dump.

The same processing will be done in a shared HFS environment. Figure 49 provides an overview of DFSMSdss logical dump processing for a dump issued from an client system in a participating group (SYSBPX sysplex group).

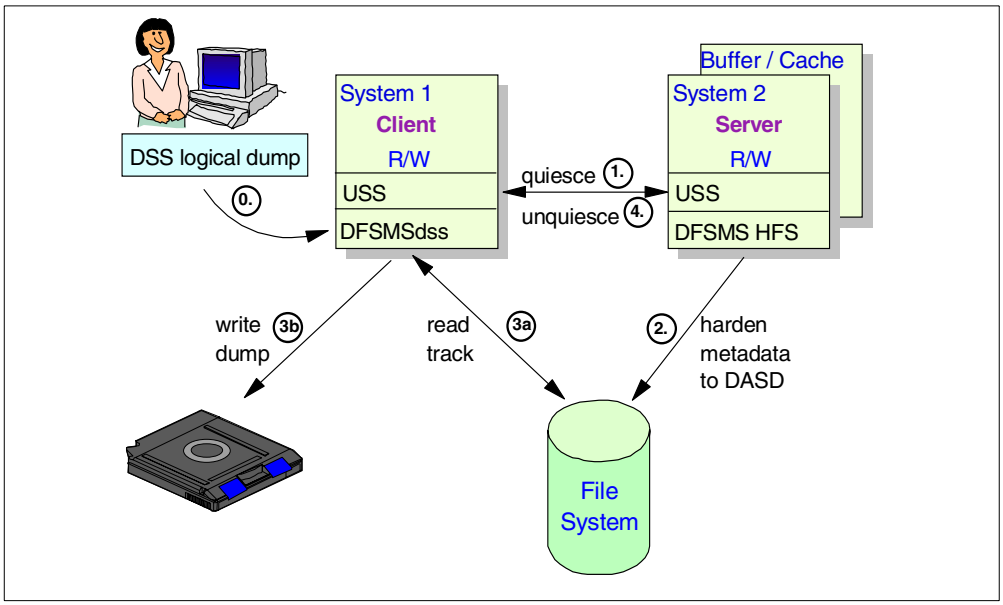


Figure 49. DFSMSdss dump in a shared-HFS environment

Refer to Figure 49 callouts:

1. DFSMSdss invokes the quiesce processing on system 1. The quiesce is propagated to system 2. It makes the files in it unavailable for use.
2. The server also performs a sync to harden the updated metadata that are located in his buffers to the HFS data set on DASD. After the file system is quiesced, system 1 will be notified and the data in it can be backed up.
3. DFSMSdss starts the I/O processing to read all tracks from the HFS data set located on DASD (3a) and writes the data into the sequential output data set on Tape or DASD (3b).
4. if DFSMSdss has completed the I/O processing then it invokes the unquiesce macro to make the file system available again and to continue the normal file system processing.

The DFSMSdss process will be finished after unquiesce processing completes successfully.

In B.5, “Shared HFS - DFSMSdss dump from client” on page 281, it shows an example of DFSMSdss logical dump processing.

Note: Currently, the file systems will also be quiesced during DFSMSdss logical dump processing if you have mounted the HFS in read-only mode.

You receive the following message if you try to dump an HFS data set when it is currently mounted R/W on another system in a non-shared environment or if you try to perform a physical dump for an HFS data set that is currently mounted:

```
ADR412E (001)-DTDSC(03), DATA SET HFS.OS390R7.SC63.O37RA1.ROOT IN CATALOG
UCAT.VSBOX01 ON VOLUME SBOX14 FAILED SERIALIZATION
```

In DFSMSdss 1.4 and prior releases, the **SHARE** keyword must be specified when dumping a mounted HFS data set. When an HFS data set is currently mounted, OMVS will have a shared SYSDSN ENQ. Before DFSMSdss 1.5, DFSMSdss obtains an exclusive SYSDSN ENQ if the SHARE keyword is not specified.

FAMS (File and Attributes Management Service) is called to perform the quiesce against the HFS data set before dumping it. The quiesce will succeed if the data set is not mounted or if it is mounted on the same system that it is being dumped from.

In DFSMSdss 1.5, DFSMSdss no longer obtains a SYSDSN ENQ, so the SHARE keyword is no longer required during logical dump. For further information on serialization of HFS data sets, see 7.1.2, “Serialization by DFSMSdss” on page 173. FAMS is no longer called to perform the quiesce.

Note

For shared HFS in OS/390 2.9 and higher:

During our tests, from time to time, we received a DFSMSdss message `'ADR412E ... FAILED SERIALIZATION'` during logical dump processing. This was because our DFSMSdss DUMP job was routed to a system which was not part of the participating group.

Please keep in mind, that you can specify `/*JOBPARM SYSF=systemname` in your job to route the job to a specific system which is part of the SYSBPX group.

For example (see Figure 72 on page 172), we submitted the job on system SC64, but the job was routed to system SC63. System SC63 did not belong to the participating group, therefore, we correctly received the ADR412E message.

5.1.1.2 Concurrent copy and virtual concurrent copy

DFSMSdss provides the concurrent copy (CC) function that, when used with supported hardware, provides point-in-time data consistency. The data is copied as if no updates have occurred. This function is invoked through the CONCURRENT keyword in the DUMP command.

If the source volume is a RAMAC Virtual Array and CONCURRENT is specified, DFSMSdss uses the Snapshot capability of the RVA to provide a function equivalent to concurrent copy. This function is called CC-compatible Snapshot — or Virtual Concurrent Copy (VCC) — and is transparent to the user.

HFS data sets can be dumped by using concurrent copy (CC or VCC).

Note: Concurrent copy requires serialization on the data being processed while the concurrent copy environment is initialized. This is the same serialization used by DFSMSdss DUMP without using the CC option. The benefit of concurrent copy is that this period of serialization is substantially reduced when compared with a non-concurrent copy operation.

For detailed information about DFSMSdss and concurrent copy, refer to:

- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Guide,* SC26-4930
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Reference*, SC26-4929

Concurrent copy (CC)

The concurrent copy function of DFSMSdss is a hardware and software solution that allows you to back up a database or any collection of data at a point-in-time, and with minimum down time, for an HFS or data base.

The system serializes access to the data being dumped or copied just long enough for the concurrent copy session to be initialized. This serialization takes a matter of seconds, unlike the quiesce and backup technique, which requires data to be unavailable for the entire duration of the dump, possibly for hours.

The copy is logically complete as soon as the concurrent copy environment is initialized. At this point, the original state of the data is “protected” by concurrent copy. After logical completion, the data is once again available for unrestricted application access. The copy is physically complete once the concurrent copy process finishes copying the data to the output device.

After concurrent copy initialization, DFSMSdss releases all the serialization it holds on the data, informs the user that the initialization is complete so that update activity may resume, and begins reading the data.

Be aware, however, that concurrent copy does not remove all data integrity exposures. For example, a DFSMSdss full-volume dump serializes the VTOC of the source volume, but does not serialize the data sets on the volume. This ensures that the existing data sets are not deleted or extended, and new data sets are not allocated. However, there is an exposure in that the data in the existing data sets can be changed. Without concurrent copy, this exposure exists for the entire duration of the dump. With concurrent copy, the exposure exists only during initialization.

Logical data set dump processing of HFS data sets, full volume, and physical data set dump operations are processed on a track-by-track basis by DFSMSdss.

Refer to the redbook *Implementing Concurrent Copy*, GG24-3990, for details on Concurrent Copy.

Virtual concurrent copy (VCC)

CC-compatible Snapshot support uses Snapshot to provide a concurrent copy-like function when the source device supports Snapshot, but does not support concurrent copy.

If you are already using concurrent copy, you do not have to make changes to your JCL to use virtual concurrent copy. To invoke VCC, you specify the CONCURRENT keyword on a DFSMSdss COPY or DUMP statement.

During CC-compatible Snapshot (VCC), data is *snapped* from the source location to an intermediate location called the working space data set, and the data is gradually copied to the target location using normal I/O methods. The operation is logically complete after the source data is *snapped* to the working space data sets and physically complete after the data is moved to the target media.

Refer to the redbook *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268, regarding additional information about VCC.

5.1.1.3 HFS and ALLDATA(*) considerations

The amount of space dumped during logical dump and restore processing is related to the high formatted frame number (HFRFN).

See 1.2, “Structure of an HFS data set” on page 2, for additional information on high formatted and high allocated differences.

Dumping without specifying ALLDATA(*)

If you do not specify ALLDATA(*) at DUMP time then DFSMSdss will only dump the storage to the high formatted value (HFRFN). On restore processing, DFSMSdss allocates the target HFS data set based on the HFRFN. This means that DFSMSdss could reduce the amount of space for an HFS data set from the allocated space to the (high) formatted space.

Figure 50 shows that the space from the top to the high formatted page of the HFS data set will be dumped and restored.

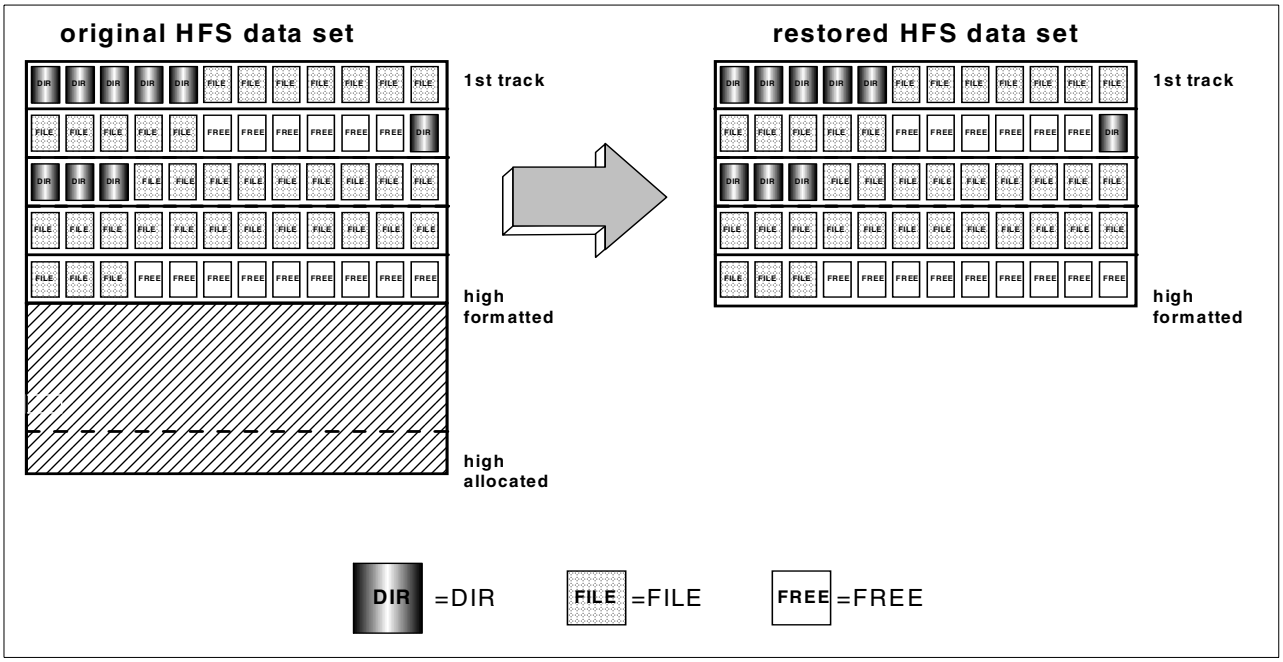


Figure 50. Logical DFSMSdss dump and restore without ALLDATA(*)

Notes:

- You cannot reduce the space below the high formatted page by using DFSMSdss DUMP/RESTORE or DFSMShsm MIGRATE/RECALL. To reduce the space of an HFS data set below the high formatted page, you must copy

files and directories individually into a new HFS data set by using UNIX commands.

- You can also use the *copytree* utility provided by OS/390 Unix System Services. *Copytree* is a utility that can run under TSO or the shell and is used to make a copy of a file hierarchy preserving all file attributes. It can be downloaded from the following *OS/390 Unix System Services* Internet page:

  `http://www.s390.ibm.com/oe/bpxa1ty2.html`

  Or, you can use:

  `http://www.s390.ibm.com/oe/`

  Select **Tools & Toys —> OS/390 Unix Tools**.

- Refer to 5.3.1, “Copytree utility” on page 127.

Dumping with specifying ALLDATA(*)
If you specify ALLDATA(*) at DUMP time then DFSMSdss will allocate the target HFS data set with the same size as the original (source) HFS data set.

Figure 51 shows a sample structure of an HFS data set if the DFSMSdss DUMP is performed with parameter ALLDATA(*). In this case, the allocated space will be preserved at restore time.

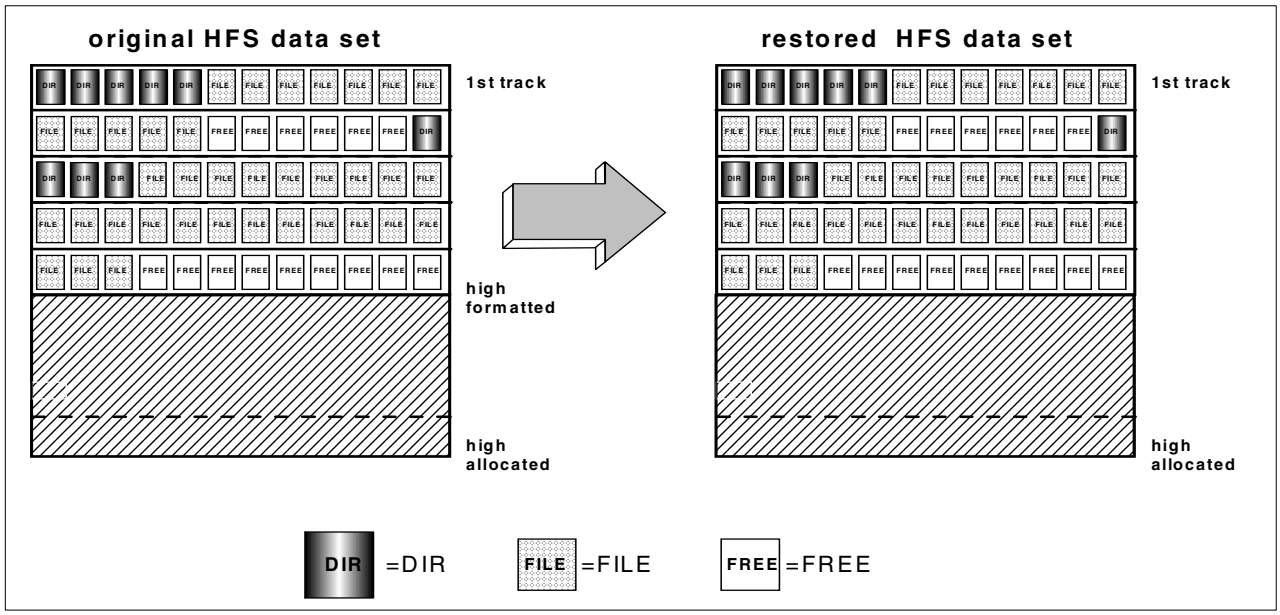


Figure 51. Logical DFSMSdss dump and restore with ALLDATA(*)

Note: DFSMSdss will not dump residual data past the high formatted page number when dumping HFS data sets, even if ALLDATA was specified during DUMP. However, the target data set will still be allocated to the high allocated page, if ALLDATA was specified during DUMP.

5.1.1.4 Logical volume dump
To perform a logical volume dump, you specify DATASET(INCLUDE(**)) with either LOGINDDNAME or LOGINDYNAM.

LOGINDDNAME identifies the input volume that contains the data sets to be dumped. LOGINDYNAM specifies that the volumes containing data sets to be dumped be dynamically allocated.

Note: The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).

You can also specify the SELECTMULTI parameter to select the method for determining how cataloged multi-volume data sets are to be selected during a logical data set dump operation. SELECTMULTI is accepted only when logical volume filtering is specified with either LOGINDDNAME or LOGINDYNAM keywords. If logical volume filtering is not used, the specification of SELECTMULTI is not accepted.

- **ALL:** Is the default, and specifies that DFSMSdss does *not* dump a multi-volume data set unless the volume list specified by LOGINDDNAME or LOGINDYNAM lists all the volumes that contain a part of the HFS data set.
- **ANY:** Specifies that DFSMSdss dump a multi-volume data set when any volume in the volume list specified by LOGINDDNAME or LOGINDYNAM contains a part of the HFS data set.
- **FIRST:** Specifies that DFSMSdss dump a multi-volume data set only when the volume list specified by LOGINDDNAME or LOGINDYNAM lists the volume that contains the first part of the HFS data set.

If either LOGINDDNAME or LOGINDYNAM is specified, DFSMSdss uses logical processing to perform the dump operation. Logical processing is also used if no input volume is specified.

A multi-volume data set that has extents on volumes not specified with LOGINDDNAME or LOGINDYNAM will not be dumped unless you specify SELECTMULTI.

5.1.1.5 Examples of logical dump processing

You can select data sets for DFSMSdss processing by filtering on criteria you specify. DFSMSdss can filter on fully or partially qualified data set names and on various data set characteristics.

The SHARE keyword is required to logically dump mounted HFS data sets in DFSMSdss releases prior to DFSMSdss 1.5. It is *no longer* required when logically dumping HFS data sets beginning with DFSMSdss release 1.5.

To dump an HFS data set while it is mounted in read/write mode you can specify:

```
DUMP DATASET(INCLUDE(hfs.data.set.name)) -
     OUTDDNAME(ddname)
```

You can dump all allocated space using the ALLDATA(*) keyword.

```
DUMP DATASET(INCLUDE(hfs.data.set.name)) -
     ALLDATA(*) -
     OUTDDNAME(ddname)
```

Using the CONCURRENT (or CC) keyword will minimize the time that the HFS data set is serialized.

```
DUMP DATASET(INCLUDE(hfs.data.set.name)) -
     CC -
     OUTDDNAME(ddname)
```

DFSMSdss can filter on fully qualified or partially qualified data set names (by using the INCLUDE or EXCLUDE keyword) and on various data set characteristics (by using the BY keyword).

The next example shows how to select all HFS data sets on one volume by using a filter on DSORG (data set organization).

```
DUMP DATASET(BY(DSORG,EQ,HFS)) -
     LOGINDYNAM(volser) -
     OUTDDNAME(ddname)
```

Or you can filter on a partially qualified data set name. In the following sample, all data sets with a high level qualifier of OMVS and with a qualifier starting with HFS will be selected for dump processing.

```
DUMP DATASET(INCLUDE(OMVS.**.HFS*.**)) -
     OUTDDNAME(ddname)
```

The STORGRP parameter specifies that all of the online volumes in the s*torage group* are dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing.

This example can be used to dump all data sets in a storage group.

```
DUMP DATASET(INCLUDE(**)) -
     STORGRP(sgname)) -
     OUTDDNAME(ddname)
```

This example shows how to perform a **logical volume dump**.

```
DUMP DATASET(INCLUDE(**)) -
     LOGINDYNAM((volser1),(volser2),(volser3)) -
     OUTDDNAME(ddname)
```

Or, for multi-volume processing, you can specify the **SELECTMULTI** parameter.

```
DUMP DATASET(INCLUDE(**)) -
     LOGINDYNAM(volser) SELECTMULTI(xxx) -
     OUTDDNAME(ddname)
```

Information about the DFSMSdss multi-volume processing is also provided in 5.5, “Increasing the size of an HFS data set” on page 132.

See B.2, “Sample DFSMSdss jobs” on page 262 for sample JCL.

5.1.1.6 Physical dump

Physical dump does not use the quiesce service to write cached data to disk, and it is not recommended as a way to back up HFS data sets which are mounted read/write at the same time on several systems. The SHARE keyword should never be specified during a physical dump of an HFS. Since the SHARE keyword applies to the SYSDSN ENQ, it does not provide protection against updates during dump.

If SHARE or TOL(ENQF) is specified during a physical dump, then the internal control information and data inside the HFS can change during the dump. This can result in a dump data set that contains a broken HFS data set. This data set may not be usable after it has been restored.

If you must physically dump an HFS that is in use, TOL(ENQF) should be used instead of SHARE. At least, with TOL(ENQF) the user will receive a return code of four along with a warning message if adequate serialization was not provided during dump.

Important information

A physical dump of HFS data sets which are mounted R/W is *not* recommended, because quiesce is not available during physical dump.

Using SHARE or TOL(ENQF) can result in a dump data set which contains a broken HFS data set, and it may not be usable after it has been restored.

For a physical dump of HFS data sets, all of the allocated space is always dumped, regardless of the ALLDATA keyword (see Figure 52).

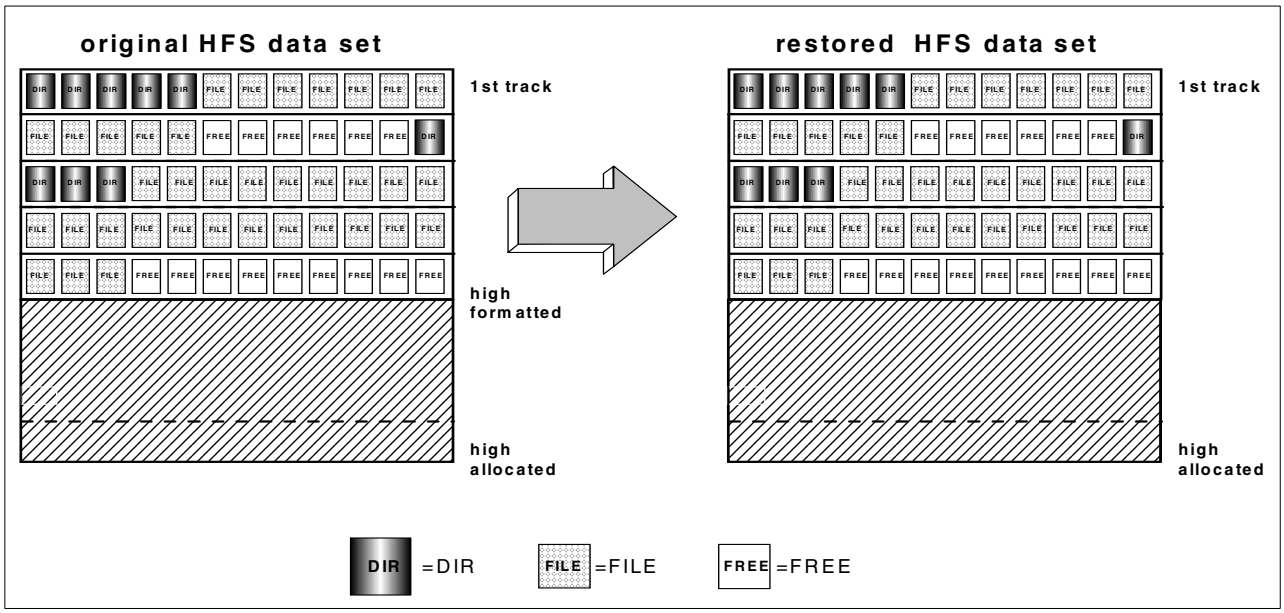


Figure 52. Physical DFSMSdss dump and restore

Note: With APAR OW39883, DFSMSdss has been changed to get an exclusive SYSDSN ENQ during physical data set DUMP of PDSE or HFS data sets unless the SHARE keyword is specified.

5.1.1.7 Sample of physical data set dump processing

If you specify INDYNAM, DFSMSdss will dynamically allocate the volume for a *physical dump* of a data set.

You can use the following statements to perform a DFSMSdss physical dump while an HFS is *not* mounted:

```
DUMP INDYNAM(volser) -
     DATASET(INCLUDE(original.hfs.dsname)) -
     OUTDDNAME(ddname)
```

See B.2, “Sample DFSMSdss jobs” on page 262 for sample JCL.

5.1.1.8 Sample of physical full volume dump

To perform a physical volume dump, specify the DUMP command with INDDNAME or INDYNAM and OUTDDNAME. Because FULL is the default keyword for the DUMP command, you need not specify it. Unallocated tracks are not dumped.

The following example shows how you can specify the DUMP command to physically back up a volume.

```
DUMP INDDNAME(ddname) OUTDDNAME(ddname)
```

Note: DFSMSdss full-volume dump serializes the VTOC of the source volume, but does not serialize the data sets on the volume. This ensures that the existing

data sets are not deleted or extended, and new data sets are not allocated. However, there is an exposure in that the data in the existing data sets (like HFS data sets) can be changed.

5.1.2 Restore processing

You can use DFSMSdss to restore HFS data sets to DASD volumes from DFSMSdss-produced dump volumes. You can restore HFS data sets to the same or a different device type if you have performed a logical dump.

DFSMSdss distinguishes between:

- Logical Data Set Restore

  A logical data set restore is performed if you are restoring from a volume created with a logical dump operation and if you specified the DATASET keyword.

- Physical Data Set Restore

  A physical data set restore is done if you are restoring from a **dump volume** created by physical dump processing and you specified the DATASET keyword.

  If the dump volumes resulted from a physical **data set dump** operation, you must do a physical data set restore.

- Volume Restore

  You can recover a volume or ranges of tracks from a full-volume dump operation.

You can also:

- Rename a data set during a restore
- Replace an existing data set
- Change either the entire name or part of the name
- Create a new data set with a new name instead of replacing the original data set on a DASD volume

During a restore operation, the data is processed the same way it was dumped because physical and logical dump tapes have different formats. If a data set is dumped logically, it is restored logically; If it is dumped physically, it is restored physically. A data set restore operation from a full volume dump is a physical data set restore operation.

You cannot restore to an HFS data set when it is currently mounted. You must first unmount the HFS data set before you can restore (REPLACE) to it. Otherwise, you will receive the following message if you try to restore an HFS data set with the REPLACE parameter when it is currently mounted:

```
ADR412E (001)-DYNA (01), DATA SET OMVS.STYRES1.HFS4 FAILED SERIALIZATION
```

5.1.2.1 Sample of logical restore processing

Here is an example of how to restore an HFS data set to a newly created HFS data set with a new name:

```
RESTORE INDD(ddname) -
   DATASET(INCLUDE(original.hfs,dsname)) -
   RENAMEU(original.hfs.dsname,target.hfs.dsname)
```

Note: You cannot RENAME the target data set and REPLACE it in the same DFSMSdss step. REPLACE only works if the data set is not being renamed. If you need to do this, you must insert a step before the restore. For example, you can use an IDCAMS DELETE, or IEFBR14, step to delete the target data set which will be renamed in the subsequent DFSMSdss RESTORE step.

See B.2, “Sample DFSMSdss jobs” on page 262 for sample JCL.

5.1.2.2 Sample of physical restore processing

This shows the DFSMSdss statements to restore an HFS data set which was dumped physically:

```
RESTORE INDD(ddname) -
   OUTDD(ddname) -
   DATASET(INCLUDE(original.hfs.dsname)) -
   RENAMEU(original.hfs.dsname,target.hfs.dsname)
```

Note: OUTDDNAME or OUTDYNAM is required for a physical restore, even for SMS managed data. They are optional for a logical restore operation, except:

- For multi-volume data sets that are preallocated on volumes that are different from the original source volumes; or
- When the original source volume is not available, and the restored data set is not going to be SMS managed.

See B.2, “Sample DFSMSdss jobs” on page 262 for sample JCL.

5.2 DFSMShsm backup and migration

This section explains DFSMShsm backup and migration procedures.

IBM offers two program products that perform backups and maintains an inventory of their attributes. They are the DFSMS/MVS Hierarchical Storage Manager (the OS/390 optional feature DFSMShsm) and Tivoli Storage Manager, formerly known as ADSTAR Distributed Storage Manager (ADSM). Tivoli Storage Manager is a separately-priced program product. See Chapter 6, “Tivoli Storage Manager” on page 153 for more information regarding TSM.

5.2.1 Common DFSMShsm functions

DFSMShsm provides automatic backup facilities for HFS data sets. You can back up HFS data sets periodically. The data sets can be restored if necessary. DFSMShsm is also used for migrating and recalling unmounted file systems.

5.2.1.1 Availability management

Availability management is the function of the DFSMShsm program that:

- Makes daily incremental backup copies of changed data sets

- Makes periodic dump copies of the DFSMShsm-managed and ML1 volumes

For example, availability management makes it possible for you to:

- Automatically make backup copies of individual changed data sets on DFSMShsm-managed volumes. This is known as incremental backup. You can specify how frequently to back up data sets on a data set basis for SMS managed data sets.
- Automatically make dump copies of DFSMShsm-managed volumes on a specified schedule for day and time. You can dump different groups of volumes on different days with different periods for the number of days between dumps.
- By command, back up user data sets.

5.2.1.2 Space management

Space management is the DFSMShsm program function that you use to ensure that you have space available on your DASD volumes to allocate new data sets or to extend old ones. You can make the space available by:

- Deleting data sets that have outlived their usefulness
- Removing unused allocated space from data sets
- Moving low-activity data sets from level 0 volumes to other DASD or tape volumes
- Returning the moved data sets to the level 0 devices when the data sets are needed

See 5.2.4, “Migrating and recalling an HFS data set” on page 125 for more information regarding HFS data sets and DFSMShsm space management.

5.2.1.3 Backup function

Backup is the process of copying a data set from a level 0 or an ML1 volume to a daily backup volume. This copy is called a backup version. The purpose of backup is to have copies of data sets in case something happens to the original data sets. The difference between dump and backup is that the dump function backs up the entire allocated space on a volume, whereas the DFSMShsm backup function backs up individual data sets.

DFSMShsm can create backup versions of data sets either automatically or through use of the BACKDS or HBACKDS commands. DFSMShsm automatically creates backup versions of data sets on specified days beginning at a specified time of day. The data sets must meet eligibility criteria and must be on DFSMShsm-managed volumes that have been designated for automatic backup.

The backup function is a **data-set-level function** when DFSMShsm is processing **SMS managed** volumes. That is, the management class attributes define how the data set is treated for creation and retention of backup versions.

DFSMShsm automatically backs up data on system-managed volumes if the storage group definition for the volumes is AUTO BACKUP=Y and the management class definitions for the individual data set is AUTO BACKUP=Y.

For **non-SMS managed** volumes, backup function is a volume-basis operation. The processing of non-SMS managed volumes is the same as that for SMS

managed volumes with few exceptions. No request is sent to DFSMSdfp for the management classes and storage classes.

Therefore, the FREQUENCY and VERSION parameters of the SETSYS command control how often data sets are backed up and how many versions are kept for all of the non-SMS managed data sets. A non-SMS managed volume can contain uncataloged data sets that can be backed up

5.2.1.4 Dump function

Dump is the process of copying all data from a DASD volume to dump tape volumes. Full-volume dump is an extension of DFSMShsm's availability management that invokes DFSMSdss through the DFSMSdss application interface. Full-volume dump backs up the entire allocated space of DFSMShsm-managed DASD volumes and ML1 volumes either automatically or by command.

5.2.1.5 Recovery and restore of data sets

Recovery and restore are processes that are requested only by command, not automatically, for backed up data sets. Recovery is the process of retrieving a full-volume dump or a backup version of a data set or a volume. Restore signifies that DFSMSdss is used to retrieve dumped data from dump volumes. You can use restore or recover processing to:

- Recover a data set that has been lost or damaged
- Recover an earlier version of the data set without deleting the current version
- Restore a volume from a full-volume dump and update the volume from later incremental backup versions
- Restore a data set from a dump copy
- Restore a volume from a full-volume dump
- Recover a volume from DFSMShsm backup versions

DFSMShsm volume recovery can use incremental backups or full-volume dumps or both. One DFSMShsm RECOVER command can be used to request both a volume restore and an incremental volume recovery.

5.2.1.6 Dump processing

Automatic dump is a **volume function**; the attributes that govern it are taken from the storage group. DFSMShsm dumps the volumes that have the AUTO DUMP attribute = Y defined in their storage groups.

DFSMShsm always allocates the source and dump volumes for the full-volume dump before it invokes DFSMSdss. DFSMSdss performs only volume level serialization during full-volume dump processing. Because DFSMShsm does not perform any data set serialization during full-volume dump processing, activity against a volume being dumped should be kept at an absolute minimum during the full-volume dump process, or errors occur.

DFSMShsm full-volume dump processing — which results in a DFSMSdss physical volume dump — means that, for an HFS data set that is mounted read/write, the HFS will not be quiesced and will not be serialized during DFSMShsm dump processing. Also, like the DFSMSdss physical dump

processing, this can result in a dump data set which contains a broken HFS data set. This data set may not be usable after it has been restored.

For additional information regarding HFS serialization, refer to:

- Section 5.1, “DFSMSdss dump and restore” on page 103
- Section 7.1, “Serialization considerations” on page 170

For additional information concerning DFSMSdss full-volume dump, refer to:

- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Guide,* SC26-4930
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Reference,* SC26-4929

5.2.1.7 SMS Storage groups and management classes

DFSMShsm is concerned mainly with DASD storage groups and with management classes. DFSMShsm manages each data set on an SMS managed volume according to the management class attributes associated with the data set.

DASD storage groups allow you to pool volumes for the purpose of defining how and whether:

- Automatic dumps are performed on the volumes and to what dump classes
- Automatic backups are performed on the volumes

  The backup function operates at a data set level, whereas the dump function backs up the entire allocated space on the volume.

Management classes allow you to group data sets logically and specify how they will be managed by DFSMShsm space management and availability management.

5.2.2 Backup processing

Unlike other non-VSAM data sets that can be opened and closed repeatedly throughout the day, some HFS file systems are often mounted for several days or weeks at a time, with the individual file members inside opened as needed. Normally, DFSMShsm's automatic backup (AUTOBACKUP) processes HFS file systems at most once per mount, so an HFS file system mounted for a week would only have one backup taken for that week. For some applications, that may not be frequent enough. Fortunately, DFSMShsm provides some alternatives to ensure that backups are taken more frequently.

- You can defined an SMS managed storage group with guaranteed backup frequency (GBF). For example, if GBF=3 days, then if a backup has not been taken for a particular data set in the last three days, a fresh backup is taken, whether the file has been updated or not. Since this applies to all data sets on a storage group, you can place your HFS data sets into a unique storage group with a specification of GBF=1, so as not to affect other types of data.

  For non-SMS managed storage, you can specify only one backup frequency for all volumes processed by any one processor. For the example, SETSYS FREQUENCY(1) defines that a backup should be taken every day.You can, however, change the backup frequency of individual non-SMS managed data sets with the (H)ALTERDS command.

- Backups once a day may not be frequent enough. DFSMShsm provides commands to invoke backups to be taken, independent of the standard autobackup cycle and window. You can use the BACKVOL TOTAL command to back up all the HFS data sets on a single DASD volume, a list of DASD volumes, a single storage group, or a list of storage groups. This command can be invoked from a job scheduling package such as OPC, or console automation package, such as Tivoli Netview/390.

  For example, in an SMS managed environment, you can issue the following DFSMShsm command to perform backup processing for an individual SMS storage group:

  ```
  BACKVOL STORGRP(xx) TOTAL
  ```

  Or, for a specific SMS managed volume:

  ```
  BACKVOL VOLUMES(volser) TOTAL
  ```

  In a non-SMS managed environment you can also specify option VOLUME(...) to backup all data sets for a specific volume:

  ```
  BACKVOL VOLUMES(volser) UNIT(3390) TOTAL
  ```

  You can submit the job or command multiple times per day to get more than one backup per day.

- If the HFS file systems are intermixed on the DASD volumes with other data set types, it might be desirable to back up the HFS individually. You can use the DFSMShsm command BACKDS to back up a single data set, or a set of data sets that match a particular mask filter. The DFSMShsm batch program ARCINBAK can be used to back up a list of data sets. ARCINBAK supports JCL backward reference and variable substitution.

  DFSMShsm also provides ABACKUP, a means to identify which HFS files systems are part of a single aggregate list, and to back these up as a single entity. Both the BACKDS and ABACKUP commands can also be invoked from job scheduling or console automation software.

- If the application was developed in-house, it can be modified to perform the backups internally. This might allow it to perform its own quiesce process, or coordinate time-stamps with its own transactional log. DFSMShsm provides the ARCHBACK assembler macro interface for this purpose.

In a OS/390 2.8 (or below) or in a non shared HFS sysplex environment if an HFS file is mounted for read/write to a single MVS image, it can be only be backed up by DFSMShsm from the MVS image that has it mounted. For automatic backup, you may need to designate host affinity by specifying a system name associated with AUTOBACKUP for each storage group (SMS managed). For command-initiated backups, you may need to ensure that the commands or batch jobs are issued to the correct MVS image. See also 5.2.2.2, “DFSMShsm multi-system considerations (OS/390 2.8 or below)” on page 121 for additional information.

If the file system is mounted as read-only or if it resides in a shared HFS sysplex environment (participating group in OS/390 2.9 or above), then it can be dumped from any system that has access to it. See also 5.2.2.3, “DFSMShsm multi-system considerations (OS/390 2.9)” on page 122.

Since DFSMShsm calls DFSMSdss to perform the backup or migrate the same restriction applies for DFSMShsm processing as we have for DFSMSdss logical dump processing.

Important information

You can backup an HFS data set from any system in participating group in a shared HFS sysplex (OS/390 2.9 and higher).

In a pre OS/390 2.9 or non-shared environment, you must perform the backup on the *same system* on which the HFS is currently mounted read/write.

For backup processing, DFSMShsm issues a DFSMSdss logical data set dump command. The DFSMSdss logical dump process itself provides the quiesce function which prevents the HFS data set from being updated at the time of the DFSMSdss logical dump (DFSMShsm backup) processing.

A DFSMShsm backup of an HFS data set results in the following DFSMSdss control statements:

```
DUMP DATASET(INCLUDE(data.set.name)) -
      OUTDDNAME(ddname) CANCELERROR OPTIMIZE(2)
```

Our test has shown that an HFS data set will be quiesced correctly during the DFSMShsm backup/DFSMSdss logical dump processing.

We issued a DFSMShsm HBACKDS command for a root HFS. The command results are shown following the DFSMSdss command:

```
DUMP DATASET(INCLUDE(OMVS.ROOTY)) -
     OUTDDNAME(SYS02240) CANCELERROR OPTIMIZE(2)
```

A D OMVS,F command shows (pre-OS/390 2.9) that the root HFS was quiesced from DFHSM.

```
BPXO044I 02.56.23 DISPLAY OMVS 513
OMVS     000E ACTIVE          OMVS=(01)
TYPENAME   DEVICE ----------STATUS----------- MODE QJOBNAME  QPID
HFS             4 ACTIVE                      RDWR
  NAME=IMW.SIMWHFS
  PATH=/usr/lpp/internet
HFS             3 ACTIVE                      RDWR
  NAME=OMVS.HFWL141.DATA
  PATH=/usr/lpp/fw/fwdata
HFS             2 ACTIVE                      RDWR
  NAME=OMVS.HFWL141.HFSPROD
  PATH=/usr/lpp/fw
HFS             1 QUIESCED                    RDWR DFHSM    838860815
  NAME=OMVS.ROOTY
  PATH=/
```

Later, when the backup was finished, the file system was active, as shown in below.

```
RESPONSE=MCECEBC
 BPXO044I 08.35.54 DISPLAY OMVS 354
 OMVS     000E ACTIVE          OMVS=(01)
 TYPENAME   DEVICE ----------STATUS----------- MODE QJOBNAME  QPID
 HFS             4 ACTIVE                      RDWR
   NAME=IMW.SIMWHFS
   PATH=/usr/lpp/internet
 HFS             3 ACTIVE                      RDWR
   NAME=OMVS.HFWL141.DATA
   PATH=/usr/lpp/fw/fwdata
 HFS             2 ACTIVE                      RDWR
   NAME=OMVS.HFWL141.HFSPROD
   PATH=/usr/lpp/fw
 HFS             1 ACTIVE                      RDWR
   NAME=OMVS.ROOTY
   PATH=/
```

For additional information regarding HFS serialization, refer to:

- Section 5.1, “DFSMSdss dump and restore” on page 103
- Section 7.1, “Serialization considerations” on page 170

5.2.2.1 DFSMShsm requirements for OS/390 USS access

If you use DFSMShsm, you must define a user ID for the DFSMShsm address space. For DFSMShsm to access the HFS data sets, it must run under a user ID that is set up for access to a OS/390 UNIX system:

- The default group for the DFSMShsm user ID must have an OMVS segment defined and a group ID associated with it.
- The home directory should be the root file system.
- If you use DFSMSdss to dump or restore an active HFS, the user ID used must be set up to have superuser authority (UID of 0). Superuser authority is required to quiesce and unquiesce a file system. If the HFS is not mounted, then it is treated as an MVS data set, and the user ID must have read (dump) or update (restore) authority.

  Note: When the DFSMSdss 1.5 HFS support was shipped, users had to run from an ID which had superuser authority. APAR OW37927 has lifted the restriction that the QUEISCE macro requires users to have superuser authority. So, DFSMShsm no longer needs to be an authorized user.

5.2.2.2 DFSMShsm multi-system considerations (OS/390 2.8 or below)

In a OS/390 2.8 (or below) or in a *non-shared* HFS environment, if the file system being dumped by DFSMShsm is currently mounted as read/write, then this file system can only be dumped from the system on which it is mounted.

Figure 53 shows a non shared environment. System 1 has mounted the file system in mode read-write. DFSMShsm backup processing can be run only on system1.

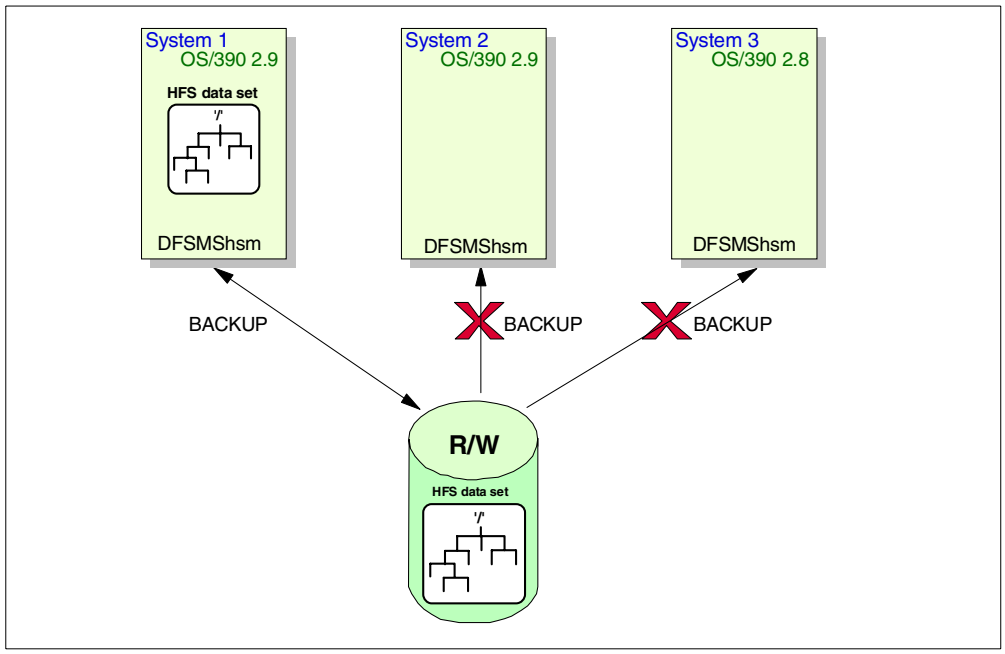


Figure 53. DFSMShsm multi-systems backup in OS/390 2.8 or below

Important Information

DFSMShsm AUTOBACKUP:

You must perform the DFSMShsm backup (phase 4 below) from the system that has mounted the HFS if the HFS data set is currently mounted as read/write.

There are four phases to DFSMShsm automatic backup:

1. Backing up the DFSMShsm control data sets
2. Moving backup versions from DASD to tape
3. Backing up migrated data sets
4. Backing up DFSMShsm managed volumes

In a multi-host configuration, the primary host performs the first three phases. The fourth phase can be performed on any host.

For more information regarding DFSMSdss, refer to:

- Section 5.1, “DFSMSdss dump and restore” on page 103
- Section 7.1, “Serialization considerations” on page 170

5.2.2.3 DFSMShsm multi-system considerations (OS/390 2.9)

In a *shared* HFS Sysplex environment in OS/390 2.9 and higher, if all systems are part of the same SYSBPX Sysplex (participating group), the DFSMShsm backup can be performed from any system that has access to the file system.

Figure 54 shows that system1 and system 2 are sharing the file system in read/write mode. The DFSMShsm backup can be done on system 1 or system 2. System 3, which is not a member of the SYSBPX group, cannot perform the backup.

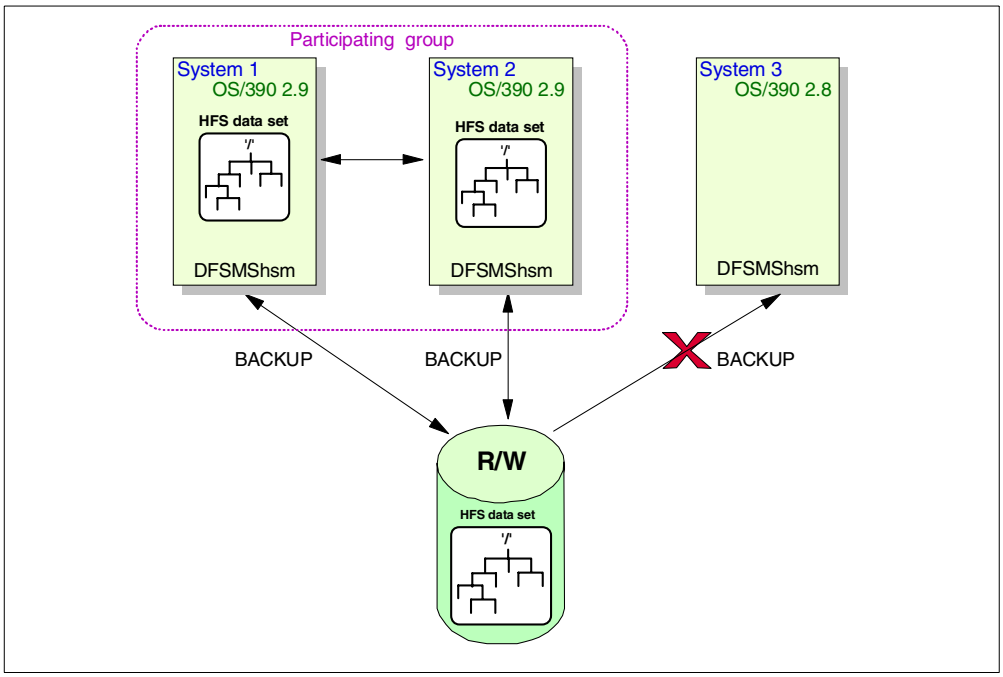


Figure 54. DFSMShsm multi-systems backup in OS/390 2.9

For more information regarding DFSMSdss, see also:

- Section 5.1, “DFSMSdss dump and restore” on page 103
- Section 7.1.2, “Serialization by DFSMSdss” on page 173

5.2.3 Recovery and restore of HFS data sets

The recovery methodology for HFS data sets used by DFSMShsm is the same as used for other non-VSAM data sets.

You use the recovery and restore process to:

- Recover a data set that has been lost or damaged.
- Access an earlier version of the data set without deleting the current version.
- Restore a data set from a dump copy.
- Restore a volume from a dump copy and update it from incremental backup versions.
- Restore a volume from a full-volume dump.
- Recover a volume from DFSMShsm backup versions.

The recovered backup version can replace an existing data set, or you can rename the recovered copy so it exists in parallel with the current data set. You can recover or restore a data set or volume only by specifying the RECOVER command.

You can use RECOVER or HRECOVER commands to recover a data set from a DFSMShsm backup version or dump copy. The HRECOVER command can be used by any user; The RECOVER command can be used by DFSMShsm-authorized users.

DFSMShsm's recovery process involves five steps for **SMS managed** data sets:

1. Determine if the data set being recovered currently exists on a level 0 volume.
2. Select a target volume.
3. Determine the storage class, management class, and data class to be associated with the data set.
4. Determine whether to recover a backup version or a dump copy.
5. Recover the data set to the target volume.

The recovery action taken depends on whether a DFSMShsm backup version or a dump copy is used to recover the data set. If you are recovering an existing data set, you must specify either REPLACE or NEWNAME. If you specify REPLACE, DFSMShsm replaces the existing data set with the recovered data set. If you specify NEWNAME, DFSMShsm recovers the data set with the NEWNAME data set name.

Note:

The HFS data set to be recovered cannot be mounted at recovery time.

Part 3, *How to Control Availability Management*, in the book *DFSMS/MVS V1R5 DFSMShsm Storage Administration Guide*, SH21-1076, provides further information about DFSMShsm recovery for SMS managed and non-SMS managed data sets.

For example, you can use following TSO DFSMShsm command to recover an non-existing SMS managed HFS data set:

```
HRECOVER 'hfs.data.set.name'
```

Only the data set name is a required parameter. However, you can specify parameters like GENERATION, DATE, VERSION, FROMVOLUME to recover a particular backup version and other parameters like NEWNAME, REPLACE to affect the data set allocation.

For **non-SMS managed** HFS data sets you should also specify UNIT and TOVOLUME parameters.

```
HRECOVER 'hfs.data.set.name.nonsms' TOVOLUME(volser) UNIT(3390)
```

These parameters are available for both the RECOVER and HRECOVER commands. However, the RECOVER command provides additional parameters not available in the user command to perform a more selective recovery for data sets or volumes.

For further information regarding the command syntax and additional parameters, please see:

- *DFSMS/MVS V1R5 DFSMShsm Managing Your Own Data*, SH21-1077
- *DFSMS/MVS V1R5 DFSMShsm Storage Administration Reference*, SH21-1075

5.2.3.1 Examples how to code the RECOVER command

In this example, a backup version or dump copy of a cataloged non-SMS data set (whichever is the more recent) is recovered to the TOVOLUME specified.

```
RECOVER hfs.data.set.name TOVOLUME(volser) UNIT(3390)
```

In this example, a next-to-latest backup generation of a cataloged data set is recovered.

```
RECOVER hfs.data.set.name GENERATION(1)
```

In this example, the oldest backup version of a cataloged data set is recovered.

```
RECOVER hfs.data.set.name VERSION(1)
```

In this example, a data set is recovered to the volume the data set is currently cataloged on, and it replaces a data set having the same name. DFSMShsm deletes the original data set.

```
RECOVER hfs.data.set.name REPLACE
```

In this example, an entire volume is recovered. It is assumed that the volume is current as of the date specified. For example, the volume has been restored with a DFSMSdss dump tape made on that date. Data sets are recovered to the volume only if a backup version was created on or after the specified date.

```
RECOVER * TOVOLUME(volser) UNIT(3390) DATE(1/12/99)
```

In this example, a lost volume is to be recovered to a DASD volume with a different volser. The TARGETVOLUME parameter saves you the step of "clipping" a volume to the lost volser.

```
RECOVER * TOVOLUME(volser) UNIT(3390) TARGETVOLUME(volser)  +
         FROMDUMP(DUMPVOLUME(volser) APPLYINCREMENTAL)
```

5.2.4 Migrating and recalling an HFS data set

If a file system is unmounted and remains so for a predetermined time, DFSMShsm can migrate it to a lower priority storage medium.

DFSMShsm invokes DFSMSdss to perform the data movement for the migration. DFSMSdss issues a logical data set dump with the following control statements:

```
DUMP DATASET(INCLUDE(hfs.dsname)) -
      OUTDDNAME(ddname) CANCELERROR OPTIMIZE(2)
```

DFSMS/MVS automatically recalls a migrated file system from the migration volume if a mount command is issued for the file system. See 4.2, “Mounting an HFS” on page 91 for additional information on mounting an HFS.

Part 2, *How to Control Space Management*, in the book *DFSMS/MVS V1R5 DFSMShsm Storage Administration Guide*, SH21-1076, provides further information about DFSMShsm migration for SMS managed and non-SMS managed data sets.

It is up to the installation to ensure that all HFS data sets specified on MOUNT statements in the BPXPRMxx PARMLIB member are available at IPL time. If an HFS data set is migrated by DFSMShsm, then the initialization of OMVS and DFSMShsm will deadlock and your system will have neither kernel nor DFSMShsm services available.

From a performance point of view, if you plan to migrate HFS data sets, migrate them only to level 1 (DASD) storage. Recalling an HFS data set that was migrated to tape could adversely affect performance because of the time required to physically mount the volume.

If your HFS data sets reside on RVA volumes, which are already compressed by more than the factor that DFSMShsm or the CPU can compress, we suggest that you eliminate automatic migration to ML1. You can leave data for longer periods of time on the primary volumes or save some space and then migrate directly to ML2.

Another side effect could be that the size of the HFS data set would be reduced. As explained in 1.2, “Structure of an HFS data set” on page 2, and 5.1, “DFSMSdss dump and restore” on page 103, an HFS data set can be reduced from the high allocated space to the high formatted space.

5.3 Recovering files and directories

If you detect a problem in a file system, we suggest that you perform the following steps to recover your data as best as possible. Note: We assume that you have taken a DFSMSdss dump or DFSMShsm backup and incremental TSM backups before the failure.

- Create a new HFS data set and mount this HFS data set to a different mount point.
- You may must unmount and then remount the affected file system (HFS data set) in read-only mode if the affected file system is flagged in error in the main control block (called the RFS). You can use the confighfs command to display the RFS error flag for a specific file system.

  Note: You may not be able to mount the file system again depending on the nature of the error (for example, if you lose a complete volume in a multi-volume HFS data set).
- Run *pax copy* function (this is a supported tool, see A.1.1, “Using pax to back up and restore files” on page 245) or *copytree* (see 5.3.1, “Copytree utility” on page 127) to copy the affected file system into the new file system (HFS data set) to salvage as many files and directories as possible. Some files could have been changed after the last backup action, so this will attempt to copy the files over the new file system.

  Notes:

  The TSM backup may not work if the directory tree is corrupted (see also 6.3, “Data management strategies” on page 163).

  Depending on the kind of damage, you may not able to salvage many files or directories. For example, if one volume of a multi-volume HFS data set is not available, the metadata pages on this volume are also not available. Therefore, USS will not be able to access many files within an HFS data set. In the worst case, you will not be able to access any files.
- Restore the file system from backup.
  - Depending on the failure, you could restore individual files and directories from TSM backups.

- If the failure does not allow you to do this, you must restore or recover the complete file system (HFS data set) from a DFSMShsm backup version or a DFSMSdss dump version. Depending on your backup strategy, you may also restore individual files and directories from TSM backups afterwards, because some files could have been backed up after the last DFSMSdss or DFSMShsm backup action.

- Now, you have recovered your broken file system (HFS data set). But the restored files and directories will only be as current as the date of the last TSM backup, DFSMShsm backup or DFSMSdss dump.

  Note: If possible, you should try to verify if the restored files and directories have been changed since the last backup action. You need to manually recreate the changes in the files and directories. A 'query filespace 'f=d' from the TSM admin client will show you the time of the last backup.

- Unmount the affected file system and rename it. You should then contact your local IBM support center so they can aid you in determining what caused the failure.

- You may wish to rename and remount the new file system to the original HFS data set name and at the original mount point.

  Note: Please keep the affected file system for further analysis by your IBM support center.

See B.6, "Recovery samples" on page 284 for sample recovery scenarios.

5.3.1 Copytree utility

OS/390 USS provides the copytree tool designed for OS/390 UNIX by IBM developers and testers. The copytree tool is helpful for copying and consistency checking on file systems. You can get the tool from the OS/390 UNIX internet pages.

Note: The OS/390 USS tools are free. There are no warranties of any kind, and there is no service or technical support available for these tools from IBM.

They can be downloaded from the following *OS/390 Unix System Services* Internet page:

`http://www.s390.ibm.com/oe/bpxa1ty2.html`

Or, you can use:

`http://www.s390.ibm.com/oe/`

Select **Tools & Toys —> OS/390 Unix Tools**.

Please see the USS Internet pages for the latest information regarding the utilities.

5.3.1.1 Copying an HFS

You can use the copytree utility to copy a file system into another file system.

A big advantage is of using copytree is that you have (re)created a file system from scratch, including the metadata. The processing will be performed on a file basis. You will see the following benefits:

- Cleanup of index records (metadata).

- Removing orphaned metadata pages and orphaned data pages.
- Defragmentation (DEFRAG) processing for the individual files in a file system. This means that all pages that belongs to an individual file are copied in ascending order on the DASD volume.
- Performance improvements, mainly for applications which are performing sequential access to the files (Also see 6.4.2, “Reorganizing HFS for performance” on page 165).

Important Information

We recommend regular use of either the supported pax USS command or the copytree utility based on the benefits described above.

Copytree is a utility that can run under TSO or the shell and is used to make a copy of a file hierarchy preserving all file attributes. pax is a USS command

Copytree replicates the source tree under the source directory within a file system to a target directory or attempts to verify the integrity of a source tree without copying it. Every attribute that can be set should be copied if you have sufficient authorization. Other features include:

- Should tolerate errors when setting target attributes with messages
- Should tolerate errors in the source tree, skipping those files
- Copies sparse files as sparse files
- Preserves file links
- Handles both symlinks and external links
- Does not cross mount points
- Runs under TSO or the shell

There are two restrictions: copytree will not handle files >1GB in TSO and it requires the REXX function package for >1GB files when run under the shell.

The syntax for the copytree utility is:

```
copytree <sourcedir> [<targetdir>]
```

`<sourcedir>` is the pathname for the source directory where the copy begins

`<targetdir>` is the pathname for the target directory. This directory must exist and must be empty. If the <targetdir> is not specified, copytree runs in a mode that checks the source file tree.

Copytree can be installed in the HFS or in a PDS to run under TSO. Download the copytree REXX program in text mode and copy it to a PDS where REXX execs can be run and/or to an HFS directory where programs can be run. Permissions should be at least read and execute; 755 is recommended.

For example: `copytree /u/nigelr2/old2 /u/nigelr2/new2`

```
NIGELR2 @ SC64:/u>copytree /u/nigelr2/old2 /u/nigelr2/new2
Copying /u/nigelr2/old2 to /u/nigelr2/new2
Scanning for file nodes...
Skipping mountpoint: /u/nigelr2/old2/..
Processing 30 nodes
Creating directories
Creating other files
Setting file attributes

******************

Copy complete.  Error count= 0
Directory errors:   0
Directories copied: 7
File errors:        0
Files copied:       21
Symlink errors:     0
Symlinks copied:    2
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
```

See also B.6, “Recovery samples” on page 284.

5.3.2 Consistency checking of a file system

The checking mode of the copytree utility can be used to verify the consistency of an HFS data set. This mode is also referred as the *checktree* function.

This mode was provided in a new version of the copytree utility which was created in March 2000. Copytree now has two execution modes:

- The original mode which copies all of the nodes from a file system to a new file system.
- A checking mode. In this mode, it does not write any files. It only verifies that it can read the first page of every file in the file system. By doing this, it implicitly verifies that the file attribute and space map information is all accessible from the HFS index. This goes most of the way towards verifying the integrity of the HFS metadata.

The checktree function also detects sparse files. If a program only writes portions of a file space, HFS will not allocate disk pages in the unwritten portions. However, if a program issues a read to the unwritten portions, HFS returns zeros rather than an indication that data does not exist. This is required for UNIX compliance.

For example, a Domino file system is not supposed to be sparse. So, if a sparse file in found in a Domino HFS, it is probably corrupt.

For example: `copytree /u/nigelr2/old2`

```
NIGELR2 @ SC64:/u>copytree /u/nigelr2/old2
Checking /u/nigelr2/old2
Scanning for file nodes...
Skipping mountpoint: /u/nigelr2/old2/..
Processing 30 nodes

******************

Check complete.  Error count= 0
Directory errors:   0
File errors:        0
Symlink errors:     0
Char-spec errors:   0
FIFO errors:        0
Sparse file count:  0
```

See B.6, “Recovery samples” on page 284.

5.4 Additional space management topics

The next sections provide information about releasing unused space in an HFS data set, and deleting or transporting an HFS data set.

5.4.1 Releasing unused space

DFSMS/MVS generally provides two ways of releasing over-allocated space.

5.4.1.1 Releasing unused space from a DASD data set using PARTREL

DADSM supports the release of unused space that is allocated. However, release of HFS data sets is **ignored**, since the DADSM PARTREL macro no longer supports HFS data sets.

5.4.1.2 Releasing space using DFSMShsm

DFSMShsm provides two functions during automatic primary space management to reduce the unused space, but only the second one is applicable for HFS data sets:

1. Under control of the management classes for the data sets, all processors release unused allocated space in physical sequential, partitioned, and extended format virtual storage access method (VSAM) KSDS data sets.

   DFSMShsm invokes the PARTREL function of DADSM. As mentioned in the previous section, the PARTREL function is **not supported** for HFS data sets.

2. Under the extent reduction function, all processors also reduce the number of extents of physical sequential, partitioned, and direct access data sets that have exceeded a specified number of extents. During the process of extent reduction, they also release any unused space in the data sets.

   DFSMShsm **migrates** the data sets that are candidates for extent reduction and immediately schedules a **recall** for those same data sets. During the migration, DFSMShsm moves only the valid data from the data sets. Therefore, when the data sets are recalled into fewer extents, they may occupy less space.

As shown in 5.2.4, “Migrating and recalling an HFS data set” on page 125, the migrate results in a logical data set dump without ALLDATA(*).

Therefore, DFSMSdss will only dump the storage to the high formatted value (HFRFN). On DFSMShsm recall or DFSMSdss restore processing, DFSMSdss allocates the target HFS data set based on the HFRFN. This means that DFSMSdss could reduce the amount of space for an HFS data set from the allocated space to the high formatted space.

Refer to 5.1.1.3, “HFS and ALLDATA(*) considerations” on page 108 for more information about ALLDATA(*).

Notes:

1. You cannot reduce the space below the high formatted page by using either DFSMSdss DUMP/RESTORE or DFSMShsm MIGRATE/RECALL. To reduce the space of an HFS data set below the high formatted page, you must copy files and directories individually into a new HFS data set by using UNIX commands.
2. You can use the *copytree* utility provided by OS/390 Unix System Services. *Copytree* is a utility that can run under TSO or the shell and is used to make a copy of a file hierarchy preserving all file attributes. See 5.3.1, “Copytree utility” on page 127.

5.4.2 Deleting or removing an HFS data set

If the file system (HFS data set) is mounted to another file system, logically unmount it using the TSO UNMOUNT command against the HFS data set containing it.

You can remove the file system in one of the following ways:

- Use the DELETE command (IDCAMS or TSO) with the SCRATCH parameter
- Execute an IEFBR14 job with DISP=(OLD,DELETE) specified for the HFS data set
- Use ISPF option 3.4, *Data Set List Utility* using line command D - Delete data set
- Use DFSMSdss DUMP processing to delete, for example, all unreferenced HFS data sets. The following sample DFSMSdss job can be used to delete all HFS data sets belonging to a specific storage group and that have a last reference date of 90 days ago.

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**) -
      BY((REFDT,LE,*,-90),(DSORG,EQ,HFS))) -
      STORGRP(OPENMVS) -
      OUTDDNAME(OUT1) -
      DELETE
/*
```

Note:

1. The job also selects all HFS data sets that do not have a reference date set, such as all HFS data sets that were never used or mounted before.

2. PARM=('TYPRUN=NORUN') bypasses the processing. It can be specified to test and verify what the DFSMSdss job will do.

DFSMShsm Space Management functions can also delete data sets, based on:

- The amount of time since the data set was created
- The amount of time since the data set was last used
- An explicit date

5.4.3 Transporting an HFS

You might also want to copy a data set to a storage medium that can be physically transported to another location. You can do that in one of the following ways:

- Use either the pax, cpio, or tar shell commands to copy the file system in tape archive (TAR) format.
- Have an authorized user logically unmount the file system, allocate an HFS data set with a different data set name, and use the DFSMSdss logical dump utility to copy the old file system to the new data set.

5.5 Increasing the size of an HFS data set

During the initial allocation of an HFS data set you specify a primary and secondary allocation quantity. Normally during initial allocation, the system allocates the first extent based on the primary allocation. The secondary allocation quantity specifies the amount of additional space to be automatically obtained if more space is needed as users add files and extend existing files.

HFS data sets can span up to 59 volumes, with up to 255 total extents for all volumes, and up to 123 extents per volume.

Note: For DFSMS/MVS 1.4 and below, a file system resides on only one volume. In DFSMS/MVS 1.5, an HFS data set can span multiple volumes.

To control or limit the size of an HFS data set, you can define it with no secondary allocation value (zero). Additional extents will not be automatically obtained. Such an HFS data set is limited to the size of the primary allocation.

Note: However, in this case, if candidate volumes are available, the HFS automatically allocates the primary allocation amount on each candidate volume as the HFS is extended to the new volume also with no secondary quantity specified.

As mentioned before, an HFS data set size increases as users add files and extend existing files. It may be necessary to increase the size of an existing HFS data set when it has run out of extents or it can outgrow the space on its volume.

In this case, the storage administrator or system programmer responsible for HFS data sets can make more space available by doing one of the following:

- Use DFSMSdss DUMP and RESTORE to
  - Move the entire full file system to another volume
  - Restore to a larger preallocated HFS data set

- Restore to a multi-volume HFS data set, if it was a single volume HFS data set before

See 5.5.1, “Increasing the file system size using DFSMSdss” on page 133 for further information.

- Add volumes to the HFS data set by using the IDCAMS ALTER ADDVOLUMES command.

  See 5.5.2, “IDCAMS ALTER to add candidate volumes” on page 137.

- Invoke the new USS confighfs command to extend the HFS.

  See 5.5.3, “USS confighfs command to change file system size” on page 139.

- Remove other data sets from the volume on which the full HFS data set resides.
- Remove files from the full file system by either deleting them or by moving them to another file system on another volume. If it is impossible to remove the chosen files from a particular directory in the file system, it may be possible to remove other files from a different directory in the same file system.
- Create a new file system on another volume and move some files from the full file system to the new file system. To avoid problems that might result from this approach, define symbolic links using the original names.

5.5.1 Increasing the file system size using DFSMSdss

Normally, these steps should be performed to increase the size of an HFS data set:

1. Have an authorized user enter a TSO UNMOUNT command to unmount the file system.
2. Use the DFSMSdss DUMP function to logically dump the old file system.
3. Delete or rename the old HFS.
4. Allocate a new single-volume or multi-volume HFS data set.
5. Use the DFSMSdss RESTORE function to restore the dumped file system into the preallocated HFS data set.
6. MOUNT the new file system.

The main information about DFSMSdss DUMP and RESTORE processing has already been provided in 5.1, “DFSMSdss dump and restore” on page 103. The following section supplies additional information for multi-volume processing.

The statements and parameters are explained in detail in:

- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Guide,* SC26-4930
- *DFSMS/MVS V1R5 DFSMSdss Storage Administration Reference,* SC26-4929

5.5.1.1 Dump processing for multi-volume data sets

An important advantage of DFSMSdss as a backup tool is that it can back up multi-volume data sets without having to specify any or all of the input volumes. If you do not specify any input volumes (you are using catalog filtering), multi-volume data sets will automatically be processed in their entirety. The catalogs are scanned to select an entire data set; That is, the data set is

processed in its entirety from all the volumes it resides on. Logical processing consolidates the extents of the data set in one dump data set for you.

Also, be aware of the DFSMSdss parameter, SELECTMULTI, if you are performing a logical dump operation. Refer to 5.1.1.4, “Logical volume dump” on page 109, for additional information about SELECTMULTI parameter.

SELECTMULTI works only for logical data set dumps. If you dump a multi-volume data set physically, you must ensure that the segments from all the volumes are dumped together. If you dump a multi-volume data set physically, it is dumped from all the volumes that are passed. The output dumped data contains a logical file for each selected volume.

A DFSMSdss logical data set dump operation attempts to ensure that all parts of a multi-volume non-VSAM data set exist. In cases where a part of the data set is missing, such as an inadvertent scratching of the VTOC entry on a volume, DFSMSdss issues an error message and stops processing the data set.

- Sample DFSMSdss DUMP job

  A logical dump statement can be specified as follows:

```
DUMP DATASET(INCLUDE(original.hfs.data.set.name)) -
     OUTDDNAME(ddname)
```

See 5.1.1, “Dump processing” on page 103, for additional information. Refer to B.3, “DFSMSdss multi-volume processing” on page 269 for sample JCL.

5.5.1.2 RESTORE processing for multi-volume data sets

Multi-volume data sets from a logical data set dump tape can be restored either to a single volume or to multiple volumes. When they are not preallocated and the specified output volumes are different from the input volumes, multi-volume data sets are restored to a single volume, space permitting.

If you are restoring a multi-volume data set from a physical dump, be sure the segments from all volumes are restored with successive RESTORE commands. Restoring a portion of a multi-volume non-VSAM data set to a preallocated data set is allowed only if the volume sequence numbers of the source and target data sets are the same.

Here are the descriptions of several parameters, their restrictions, and some sample statements:

- **MAKEMULTI parameter**

  MAKEMULTI allows DFSMSdss to convert single volume data sets into multi-volume data sets. The default is *not* to convert single volume data sets into multi-volume data sets.

  SMS managed target data sets are given a volume count (VOLCOUNT) that is either:

  - The number of SMS output volumes specified in the RESTORE command, if output volumes are specified through OUTDDNAME or OUTDYNAM.
  - The number of volumes in the target storage group, or a value of 59, whichever is less.

Note: When MAKEMULTI is specified and VOLCOUNT is also specified with an option other than VOLCOUNT(*), the VOLCOUNT option overrides MAKEMULTI.

- **VOLCOUNT parameter**

  The number of volumes allocated for HFS data sets can be changed with VOLCOUNT keyword options. The output data set must be SMS managed. Single-volume data sets can be converted to multi-volume; multi-volume data sets can be converted to single volume; or the number of volumes allocated for multi-volume data sets can be changed. The result depends on which VOLCOUNT keyword is selected, and on whether output volumes are specified.

  The following shows two possible VOLCOUNT options which can be used for HFS processing:

  - **VOLCOUNT(N(*nn*))** - *nn* represents the number of volumes to be used for SMS data set allocation. Any value between 0 and 59 may be specified.
  - **VOLCOUNT(ANY)** specifies that DFSMSdss use a maximum volume count to allocate the SMS target data set. DFSMSdss initially sets a volume count of 59 for the allocation, but it reduces the number of volumes used to the number of volumes needed to satisfy the allocation.

  Refer to *DFSMS/MVS V1R5 DFSMSdss Storage Administration Reference*, SC26-4929, for conditions and restrictions regarding the VOLCOUNT parameter and the required options.

- **RENAMEU and REPLACE parameter restrictions**

Important information

REPLACE only works if the data set is not being renamed.

REPLACE must be specified to restore preallocated data sets.

This means that you cannot perform a restore of an HFS data set to a preallocated (target) HFS data set with a different name from that of the dumped (original) HFS data set.

However, to bypass this restriction, you can perform an additional step before you restore to your final target data set.

1. Use DFSMSdss DUMP to dump the original HFS data set.
2. Rename (or delete) the original HFS data set.
3. Preallocate a larger HFS data set.
4. Run the DFSMSdss RESTORE only with option REPLACE (no RENAME nor RENAMEU statement) to restore the HFS data set with the original data set name.

However, sometimes you cannot rename the original HFS data set. For example, if the HFS is still mounted (like the root file system), you need to insert a second DFSMSdss DUMP and RESTORE job:

1. DFSMSdss DUMP of the original HFS data set.
2. DFSMSdss RESTORE with RENAMEU to the new HFS data set (which did not exist before).

Note: At this point in time, the target HFS data set has changed the name, but not the size.

3. DUMP this HFS data set again using DFSMSdss DUMP.
4. Delete the new HFS data set (created at step 2).
5. Preallocate a larger HFS data set with the *same name* the HFS data set had in step 3.
6. Run the DFSMSdss RESTORE with only the option REPLACE (no RENAME nor RENAMEU statement) to restore the HFS data set with the new (step 3) data set name.

   Note: We now have a new HFS data set with a different name and a larger size.

OUTDDNAME or OUTDYNAM is required for a logical restore operation if the multi-volume data set is preallocated on volumes that are different from the original source volumes.

- **Sample RESTORE to multi-volume using MAKEMULTI**

  The following example shows the DFSMSdss statements that are needed to restore from a single HFS data set to a multi-volume HFS data set by using parameter MAKEMULTI.

  The target HFS data set does not pre-exist, and will be allocated automatically by DFSMSdss.

```
RESTORE INDD(ddname) -
   MAKEMULTI -
   DATASET(INCLUDE(original.hfs.dsname)) -
   RENAMEU(original.hfs.dsname,target.multivol.hfs.dsname)
```

See B.3, “DFSMSdss multi-volume processing” on page 269 for sample JCL.

- **Sample RESTORE by using VOLCOUNT(N(xx))**

  This sample shows how to restore to a multi-volume HFS data set with a volume count of 2. The target HFS data set does not pre-exist. The target data set was allocated as a multi-volume HFS data set with one candidate volume:

```
RESTORE INDD(ddname) -
   VOLCOUNT(N(2)) -
   DATASET(INCLUDE(original.hfs.dsname)) -
   RENAMEU(original.hfs.dsname,target.hfs.dsname)
```

See B.3, “DFSMSdss multi-volume processing” on page 269 for sample JCL.

- **Sample RESTORE by using VOLCOUNT(ANY)**

  This example shows how to restore to a multi-volume HFS data set by using the VOLCOUNT(ANY) parameter. The target HFS data set does not exist before the job:

```
RESTORE INDD(ddname) -
   VOLCOUNT(ANY) -
   DATASET(INCLUDE(original.hfs.dsname)) -
   RENAMEU(original.hfs.dsname,target.hfs.dsname)
```

See B.3, “DFSMSdss multi-volume processing” on page 269 for sample JCL.

- **Sample RESTORE to a preallocated single or multi-volume HFS data set**

  The restore to a pre-allocated HFS data set can be done by specifying the REPLACE option on the DFSMSdss RESTORE command:

```
RESTORE INDD(ddname) -
   DATASET(INCLUDE(original.hfs.data.set.name)) -
   REPLACE
```

See B.3, “DFSMSdss multi-volume processing” on page 269 for sample JCL.

5.5.1.3 Mounting or unmounting of the root file system

Unmount the current root file and mount the new, larger data set.

Important Information

Unmounting and remounting the root file system is very disruptive.You will have to stop all UNIX System Services work and unmount any file systems that are mounted on the root file system. You will also have to stop any address spaces, for example INETD or WEBSRV, which have HFS data sets allocated.

Once you have stopped all work, unmount the root file system through the shell, or use the following TSO command:

```
UNMOUNT FILESYSTEM('old.hfs.data.set.name') IMMED
```

You must specify the IMMED keyword when unmounting the root file system. Use the following command to mount the new file system:

```
MOUNT FILESYSTEM('new.hfs.dsname') TYPE(HFS) MOUNTPOINT('/')
```

A less disruptive method for making the changes effective is to schedule the change at the next IPL, and modify BPXPRMxx to point to the new data set name.

5.5.2 IDCAMS ALTER to add candidate volumes

IDCAMS ALTER ADDVOLUMES provides the volumes to be added to the list of candidate volumes in the associated entry in the catalog.

You can use ALTER ADDVOLUMES to add candidate volumes to HFS data sets. Only nonspecific volumes can be added to SMS managed HFS data sets.

If an ALTER ADDVOLUMES is done to a data set that is already opened and allocated, the data set must be closed, reopened, and reallocated before it can extend onto the newly-added candidate volume. This means that you must unmount the HFS data set before you can add volumes.

Important Information

An UNMOUNT and MOUNT is necessary to make use of additional candidate volumes that were added before using IDCAMS ALTER ADDVOLUMES while the HFS was mounted.

Adding a non-existent volume to the list can result in an error when the data set is extended. Ensure that the volume exists and is online before attempting to extend the data set.

SMS might not use candidate volumes for which you request specific VOLSERs with the ADDVOLUMES parameter. Sometimes a user-specified VOLSER for an SMS managed data set results in an error.

To avoid candidate volume problems with SMS, you can have SMS choose the VOLSER used for a candidate volume. To do this, you can code an * for each VOLSER that you request with the ADDVOLUMES parameter. If, however, you request both specified and unspecified VOLSERs in the same command, you must enter the specified VOLSERs first in the command syntax. Space is not allocated on candidate volumes until a data set extends to the candidate volume.

The syntax is:

```
ALTER entryname ADDVOLUMES(volser [volser])
```

For example, to add two non-specific volumes for an HFS data set with the name'OMVS.STYRES1.HFS5' you can specify:

```
ALTER OMVS.STYRES1.HFS5 ADDVOLUMES(* *)
```

Or, as a batch job:

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
    ALTER OMVS.STYRES1.HFS5 ADDVOLUMES(* *)
/*
```

The IDCAMS ALTER command is described in *DFSMS/MVS V1R5 Access Method Services for ICF*, SC26-4906.

You can use the IDCAMS LISTCAT command to display the current number of candidate volumes. For example:

```
LISTCAT ENTRIES(OMVS.STYRES1.HFS5) VOLUME
```

Or, as a batch job:

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
    LISTC ENT(OMVS.STYRES1.HFS5) VOL
/*
```

The IDCAMS LISTCAT command returns the following information.

```
IDCAMS  SYSTEM SERVICES I

    LISTCAT ENTRIES(OMVS.STYRES1.HFS5) VOLUME
NONVSAM ------- OMVS.STYRES1.HFS5
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.174
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX17     DEVTYPE------X'3010200F'
       VOLSER-----------------*     DEVTYPE------X'00000000'
       VOLSER-----------------*     DEVTYPE------X'00000000'
```

The two non-specific candidate volumes which we have just added are indicated by an * instead of a VOLSER.

5.5.3 USS confighfs command to change file system size

The USS confighfs command can be used to extend an HFS on the same, or different, volume when it has run out of extents, so that the HFS can outgrow the space on its volume.

However, the extend to another volume works only if a candidate volume is available in the associated catalog entry.

To control the space usage of an HFS data set, you can define your HFS data set with no secondary allocation value. Such an HFS will not be automatically extended. But you *can* increase the size of that HFS by using confighfs.

Note: However, in this case, if candidate volumes are available, the HFS automatically allocates the primary allocation amount on each candidate volume as the HFS is extended to the new volume also with no secondary quantity specified.

In a shared HFS sysplex (OS/390 2.9 and higher), you need to issue the confighfs command from the owning system to display information about a file system.

If you issue the confighfs from a client system, you receive a message like:

```
Error issuing PFSCTL: RC=0 ERRNO=129(81) REASON=5B360105 ERRNO=129(81): HFS is not mounted.
```

This is because the confighfs command calls the PFS directly. On a client system, the PFS doesn't know anything about the file system. See 7.3, "HFS sysplex sharing (OS/390 2.9 and above)" on page 179 for additional information regarding HFS sysplex sharing and 3.3.5, "confighfs shell command" on page 68.

Here is the syntax to extend an HFS on the same volume:

```
confighfs -x size pathname
```

Or, to extend an HFS onto another volume, you can specify:

```
confighfs -xn size pathname
```

The following parameters are used in the confighfs command:

`size` This is the amount to be extended, suffixed by the extend unit of M, T, or C (for megabytes, tracks, or cylinders).

`pathname` This is a full or simple pathname to a file or directory in the file system to extend.

Both commands require superuser authority.

This example shows how to extend an HFS data set (identified by its mount point) by allocating an additional extent of 2 tracks on the same volume:

```
confighfs -x 2T /u/styres1
```

The next example will extend the HFS to the next volume. The amount of space allocated on the next volume will be 4 tracks:

```
confighfs -xn 4T /u/styres1
```

Both commands force an allocation of a new extent, but they will not change the secondary allocation quantity in the Format-1-DSCB in the VTOC. This means that the HFS data set cannot extend automatically to a new extent on the same volume. An HFS data set can be extended to a new candidate volume if a candidate volume is available. But if no volume is available, you will receive a message stating, "`No space left on device`".

Of course you can extend the HFS again by using the confighfs command, but it will not happen automatically if you do not have a secondary allocation quantity specified.

To insert a secondary allocation quantity at a later time, you can use DFSMSdss DUMP and RESTORE to a preallocated HFS data set that has a secondary allocation quantity specified.

Another important point is that candidate volumes must be available (in the associated catalog entry) for an HFS data set that you will extend to another volume by issuing the confighfs -xn command. Because an HFS data set is still an MVS data set, neither USS nor the confighfs -xn command can bypass the DFSMSdfp DADSM rules for multi-volume data sets.

Important information

The confighfs -x command will not change the secondary allocation quantity in the Format-1-DSCB in the VTOC. If your HFS data set was allocated without specifying a secondary allocation quantity, then it will never extend automatically to a new extent on the same volume.

The confighfs -xn can only extend an HFS data set to another volume if candidate volumes are available (refer to IDCAMS ALTER ADDVOLUMES).

For the following sample, we allocated a multi-volume HFS data set with the following specifications:

Primary allocation quantity: 3 tracks
Secondary allocation quantity: 0
Volume count: 3

```
//ALLOC    EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//DDNAME   DD DSN=OMVS.STYRES1.HFS4,
//            DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(TRK,(3,0,1)),
//            DSNTYPE=HFS
/*
```

We started to fill the HFS. The HFS data set will be extended automatically to the two candidate volumes. In this example, we can get to a maximum of three extents, one on each volume. Every volume contains one extent with a size of three tracks. Therefore, the total number of allocated or used tracks is nine tracks. If we reach the limit, we will receive a message like the following:

```
FSUM6237 no space on device for file "file3": EDC5133I No space left on device.
```

```
STYRES1 @ SC64:/u/styres3>ls -l
total 680
-rwxrwxrwx   1 OMVSKERN SYS1      186600 Jun 26 00:34 file1
-rwxrwxrwx   1 OMVSKERN SYS1      156240 Jun 26 00:39 file2
STYRES1 @ SC64:/u/styres3>df
Mounted on     Filesystem               Avail/Total    Files      Status
/tmp           (/TMP)                   999805/1000000 127975     Available
/u/styres3     (OMVS.STYRES1.HFS4)      144/864        4294967292 Available
STYRES1 @ SC64:/u/styres3>cp file1 file3
cp: FSUM6237 no space on device for file "file3": EDC5133I No space left on device.
STYRES1 @ SC64:/u/styres3>
```

A LISTCAT command shows that all three volumes are already used.

```
 LISTC ENT(OMVS.STYRES1.HFS4) VOL
NONVSAM ------- OMVS.STYRES1.HFS4
VOLUMES
  VOLSER------------SBOX17     DEVTYPE------X'3010200F'
  VOLSER------------SBOX16     DEVTYPE------X'3010200F'
  VOLSER------------SBOX14     DEVTYPE------X'3010200F'
```

ISPF data set information shows the current allocation of nine tracks on three extents, and that 90% is used (Refer to 5.6, “Displaying the file system size” on page 143, regarding the file size displays).

```
Data Set Name  . . . : OMVS.STYRES1.HFS4

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 9
 Storage class  . . . : OPENMVS        Allocated extents . : 3
  Volume serial . . . : SBOX17         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 90
  Record length . . . : 0              % Utilized  . . . . : 83
  Block size  . . . . : 0              Number of members . : 3
  1st extent tracks . : 3
  Secondary tracks  . : 0
  Data set name type  : HFS
```

At this point, we have two ways to extend the size of the HFS data set:

1. We can use the confighfs -x 5T /u/styres3 command to add an extent of five tracks to the HFS on the same volume: the cp file1 file3 command now completes successfully.

   The df command shows that the HFS was extended.

```
STYRES1 @ SC64:/>cd /usr/lpp/dfsms/bin
STYRES1 @ SC64:/usr/lpp/dfsms/bin>configfs -x 5T /u/styres3
STYRES1 @ SC64:/usr/lpp/dfsms/bin>cd /u/styres3
STYRES1 @ SC64:/u/styres3>cp file1 file3
STYRES1 @ SC64:/u/styres3>df
Mounted on     Filesystem               Avail/Total    Files      Status
/tmp           (/TMP)                   999805/1000000 127975     Available
/u/styres3     (OMVS.STYRES1.HFS4)      256/1344       4294967291 Available
```

2. The second way is to extend the HFS to another volume by using the command confighfs -xn 5T /u/styres3, as in the following example.

```
STYRES1 @ SC64:/u/styres3>ls -l
total 680
-rwxr-xr-x   1 OMVSKERN SYS1      186600 Jun 26 01:27 file1
-rwxr-xr-x   1 OMVSKERN SYS1      156240 Jun 26 01:27 file2
STYRES1 @ SC64:/u/styres3>df
Mounted on     Filesystem               Avail/Total    Files      Status
/tmp           (/TMP)                   999805/1000000 127975     Available
/u/styres3     (OMVS.STYRES1.HFS4)      144/864        4294967292 Available

STYRES1 @ SC64:/u/styres3>cd /usr/lpp/dfsms/bin
STYRES1 @ SC64:/usr/lpp/dfsms/bin>configfs -xn 5T /u/styres3
Error issuing PFSCTL: RC=0 ERRNO=133(85) REASON=5B27C005
ERRNO=133(85): No space available
STYRES1 @ SC64:/usr/lpp/dfsms/bin>
```

The reason code 5B27C005 refers to the fact that no space is available:

```
RSN5B27C005 (X'C005'): RSNS_DADSM_NO_SPACE_AVAIL
```

The message was issued correctly, because no candidate volumes are available at this time.

To extend the HFS, we must first add additional candidate volumes to the associated HFS catalog entry by using the IDCAMS command:

```
ALTER OMVS.STYRES1.HFS4 ADDVOLUMES
```

Note: The HFS data set must be unmounted and remounted to make use of the additional candidate volume.

```
 NONVSAM ------- OMVS.STYRES1.HFS4
VOLUMES
        VOLSER------------SBOX17     DEVTYPE------X'3010200F'
        VOLSER------------SBOX13     DEVTYPE------X'3010200F'
        VOLSER------------SBOX16     DEVTYPE------X'3010200F'
        VOLSER-----------------*      DEVTYPE------X'00000000'
```

```
STYRES1 @ SC64:/>cd /usr/lpp/dfsms/bin
STYRES1 @ SC64:/usr/lpp/dfsms/bin>confighfs -xn 5T /u/styres3
STYRES1 @ SC64:/usr/lpp/dfsms/bin>cd /u/styres3
STYRES1 @ SC64:/u/styres3>cp file1 file3
STYRES1 @ SC64:/u/styres3>df
Mounted on     Filesystem              Avail/Total    Files      Status
/tmp           (/TMP)                  999805/1000000 127975     Available
/u/styres3     (OMVS.STYRES1.HFS4)     256/1344       4294967291 Available
```

Refer to B.4.2, “Using confighfs -xn and IDCAMS ALTER ADDVOLUMES” on page 277 for additional examples.

5.6 Displaying the file system size

In the next sections, we describe the various methods available to display the size of an HFS. Keep in mind that the physical structure on DASD is based on *pages* (4,096 bytes), therefore, every file occupies at least one *page* (4,096 bytes or 8*512 *byte blocks*). See 1.2, “Structure of an HFS data set” on page 2 for more information regarding the structure.

5.6.1 Using the df UNIX command

The df UNIX system service command shows the amount of free space left in a file system, the used space, and the total amount of available space.

Note:

In DFSMS/MVS 1.5, changes to the file system will be displayed by df after the next *sync*.

Space is measured in units of *512-byte disk sectors*. If you do not give an argument, df reports space for all mounted file systems known to the system, in the following format:

- File system root
- File system name

- Space available and total space

  The total space reported is the space in the already allocated extents (primary and any already allocated secondary extents) of the HFS data set that holds this file system. Therefore, the total space may increase as new extents are allocated.

- Number of free files (inodes)

  This number is only meaningful for file systems created using DFSMS 1.3 and later. For file systems created with earlier versions of DFSMS, this number will always be 4,294,967,295.

- File system status

The following is an example of df output.

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>df
Mounted on     Filesystem                 Avail/Total    Files      Status
/tmp           (/TMP)                     999703/1000000 127962     Available
/u/guts        (OMVS.STYRES1.HFS3)        21208/21600    4294967284 Available
/usr/lpp/lotus (OMVS.DOMINO5.PROD.HFS)    28392/724320   4294966908 Available
/var           (OMVS.SC63.VAR)            12904/12960    4294967293 Available
/u             (OMVS.SC63.USERS)          37240/37440    4294967283 Available
/etc           (OMVS.SC63.ETC)            73560/76320    4294967039 Available
/              (HFS.OS390R7.SC63.O37RA1.ROOT)87984/1264320  4294949033 Available
```

df - options

-k Uses 1024-byte (1KB) units instead of the default 512-byte units when reporting space information.

-P Lists complete information on space used, in the following order:

File system name
Total space
Space used
Space free
Percentage of space used
File system root

-t Display total allocated file slots in addition to the total number of free files that are already displayed.

-v Lists more detailed information on the file system status.

File system root
File system name
Space available and total space
Number of free files (inodes)
File system status
File system type, and mode bits
File system mount parm data

For systems in a shared HFS environment, the following additional fields are displayed:

- File system ID (owner/mounted file system server)
- File system ID issuing a quiesce request

If df is issued in a non-shared HFS environment, these two fields are not relevant and will not be displayed.

Figure 55 is a sample output of a df -P command.

```
STYRES3 @ SC63:/>df -P
Filesystem         512-blocks      Used  Available  Capacity Mounted on
/TMP                  1000000       305     999695        1% /tmp
OMVS.STYRES1.HFS3       21600       392      21208        2% /u/guts
OMVS.SC63.VAR           12960        56      12904        1% /var
OMVS.SC63.USERS         37440       200      37240        1% /u
OMVS.SC63.ETC           76320      2424      73896        4% /etc
HFS.OS390R7.SC63.O37RA1.ROOT 1264320     1175416      88904       93% /
```

Figure 55. Sample output of a df -P command

We interpret the different kinds of information provided by the different methods on a sample HFS data set 'OMVS.STYRES1.HFS3'.

The sizes are displayed in blocks of 512 bytes:

- 512 blocks (total): 21,600

  => 21,600 blocks / 8 blocks/page = 2,700 pages

  => 2,700 pages / 12 pages/track = 225 tracks

  => 225 tracks / 15 tracks/cylinder = **15 cylinders**

  The total number of 512 blocks matches the amount of allocated storage. See 5.6.3, "ISPF data set information" on page 146.

- Used blocks: 392

  => 392 blocks / 8 blocks/page = **49 pages**

  The number of used blocks matches the amount of used pages displayed in *ISPF Data Set Information*.

- Available blocks: 21,208

  This value represents:

  `total blocks - (used blocks + system reserved blocks for shadow writes)`

5.6.2 Using the UNIX confighfs command

The UNIX system service command confighfs provides similar file size information.

Important Information

Currently, in a shared HFS sysplex (OS/390 2.9 and higher), you need to issue the confighfs command from the owning system to display information about a file system. If you issue the confighfs from a client system, you receive a message like:

```
Error issuing PFSCTL: RC=0 ERRNO=129(81) REASON=5B360105 ERRNO=129(81): HFS is not mounted.
```

This is because the confighfs command calls the PFS directly. On a client system, the PFS doesn't know anything about the file system.

See 7.3, "HFS sysplex sharing (OS/390 2.9 and above)" on page 179 for additional information reading HFS sysplex sharing and 3.3.5, "confighfs shell command" on page 68.

Figure 56 is a sample output from a confighfs command.

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs /u/guts
Statistics for file system OMVS.STYRES1.HFS3
( 06/18/99 2:49pm )
 File system size:_____2700
                  _10.546875(MB)
 Used pages:      ________49
                  0.19140625(MB)
 Attribute pages: _________1
                  _0.0039063(MB)
 Cached pages:    _________4
                  __0.015625(MB)
 Seq I/O reqs:    ____________________1
 Random I/O reqs: ____________________0
 Lookup hit:      ____________________2
 Lookup miss:     ____________________1
 1st page hit:    ____________________3
 1st page miss:   ____________________5
 Index new tops:  ____________________0
 Index splits:    ____________________0
 Index joins:     ____________________0
 Index read hit:  ____________________3
 Index read miss: ____________________1
 Index write hit: ____________________0
 Index write miss:____________________0
 RFS flags        ___________________82(HEX)
 RFS error flags: ____________________0(HEX)
 High foramt RFN: ___________________31(HEX)
 Member count:    _______________12544
 Sync interval:   ___________________60(seconds)
```

Figure 56. Sample output from a confighfs command

The confighfs command displays similar information:

- File system size: 2,700 pages

  2700 pages * 4096 bytes/page =11059200 bytes <=> 10.546875(MB)

- Used pages: 49 <=> 0.19140625(MB)

Note: Refer to APAR OW39886 (USS CONFIGHFS COMMAND SHOWS INCORRECT MEMBER COUNT), if your member count is incorrect (too high) as shown in this output.

5.6.3 ISPF data set information

ISPF option 3.2 and option 3.4 (option I - data set information) also shows information about the HFS data set size; See the following example.

```
Data Set Information
Command ===>

Data Set Name  . . . : OMVS.STYRES1.HFS3

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 15
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX06         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 49
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 11
  1st extent cylinders: 15
  Secondary cylinders : 15
  Data set name type  : HFS
```

The amount of *Current Allocation* and *Used pages* matches the information provided by the UNIX *df* command.

The maximum number of free files (4,294,967,284) subtracted from the maximum number of files (4,294,967,295) is equal to the *Number of members* displayed by ISPF Data Set Information. Refer to 5.6.1, “Using the df UNIX command” on page 143.

The number of members includes the name and subname directories, so this value reflects more than just the number of files in an HFS data set.

5.6.4 TSO ISHELL — file system attributes

The *TSO ISHELL File System Attributes* panel displays the file system size in blocks of 4,096 bytes (pages).

```
File System Attributes

File system name:
OMVS.STYRES1.HFS3
Mount point:
/u/guts


Status . . . . . : Available
File system type . . . : HFS
Mount mode . . . . . : R/W
Device number  . . . . : 5
Type number  . . . . . : 1
DD name  . . . . . : SYS00008
Block size . . . . . : 4096
Total blocks . . . . . : 2700
Available blocks . . . : 2651
Blocks in use  . . . . : 49
Ignore SETUID  . . . . : 0
Bypass security  . . . : 0
Mount parameter:
   SYNC(120),NOWRITEPROTECT_____________________________
   _____________________________________________________
```

For example, the total number of 4K blocks (2,700 pages) is equal to the current allocation shown by ISPF (15 cylinders). The USS confighfs command reports the same total number of pages.

5.6.5 ISMF data set information

The ISMF values for ALLOC SPACE and ALLOC USED are not useful for HFS file size calculations, because the ISMF values represent the amount of space (in kilobytes, K=1024 bytes) *on the volume*. The values are determined by converting the number of tracks listed in the VTOC to kilobytes, and rounding to the nearest kilobyte.

The values are also correct. However, ISMF shows them from a different point of view. ISMF determines the space from a volume point of view. Therefore, the space will be calculated by using the maximum track capacity of a volume. For example, for a 3390 volume, the maximum track capacity is 56,664 bytes.

```
------------------------------------------------------------------------
                                DATA SET LIST
Command ===>

Enter Line Operators below:                                Data Columns

     LINE                                        ALLOC    ALLOC   % NOT
     OPERATOR          DATA SET NAME             SPACE    USED    USED
    ---(1)----  ------------(2)------------ --(3)--  --(4)--  -(5)-
              OMVS.STYRES1.HFS3               12451      221     98
```

- ALLOC SPACE: 12,451 KB

  => 15 cylinder = 225 tracks = 225 track * 56,664 bytes/track = **12,450.6 KB**

- USED SPACE: 221 KB

  => 4 tracks * 56,664 bytes/track = 226,656 bytes / 1,024 bytes/KB = **221.3 KB**

5.7 Moving and displaying the ownership for a shared HFS

You can use the SETOMVS system command to change the options that OS/390 UNIX System Services currently is using dynamically. These options are originally set in the BPXPRMxx parmlib member at IPL time.

In OS/390 2.9, the SETOMVS command was enhanced to:

- Change or move the ownership of an HFS mounted in read-write mode in a participating (SYSBPX sysplex) group.
- Change the setting of the AUTOMOVE parameter for an mounted file system.

We used the following command to move our file system from system SC64 to system SC65:

```
SETOMVS FILESYS,FILESYSTEM='NIGELR2.TEST.HFS',SYSNAME=SC65
```

Note

To move the ownership, the affected HFS data set will be unmounted on the source system and mounted on the target system.

A D OMVS,F command shows that the file system is owned by system SC64.

```
SC64      D OMVS,F
SC64      BPXO045I 19.23.48 DISPLAY OMVS 528
          OMVS     000F ACTIVE          OMVS=(9B)
          TYPENAME   DEVICE ----------STATUS----------- MODE
          ...
          HFS            31 ACTIVE                     RDWR
          NAME=NIGELR2.TEST.HFS
          PATH=/u/guts
          OWNER=SC64     AUTOMOVE=Y CLIENT=N
          ...
```

The corresponding messages shows, that the HFS data set is unmounted on SC64 and re-mounted on SC65.

```
SC64      SETOMVS FILESYS,FILESYSTEM='NIGELR2.TEST.HFS',SYSNAME=SC65
SC64      IEF196I IGD104I NIGELR2.TEST.HFS                              RETAINED,
SC64      IEF196I DDNAME=SYS00082
SC65      IGD103I SMS ALLOCATED TO DDNAME SYS00146
SC64      BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
```

You can also use the ISHELL to display the owner of an file system:

—> File_systems
—> 1. Mount table...
—> A=Attributes

```
File System Attributes

File system name:
NIGELR2.TEST.HFS
Mount point:
/u/guts
                                                  More:     +

Status . . . . . . . . : Available
File system type . . . : HFS
Mount mode . . . . . . : R/W
Device number  . . . . : 31
Type number  . . . . . : 1
DD name  . . . . . . . :
Block size . . . . . . : 4096
Total blocks . . . . . : 361080
Available blocks . . . : 32754
Blocks in use  . . . . : 328142

Ignore SETUID  . . . . : 0
Bypass security  . . . : 0
Automove . . . . . . . : Yes
Owning system  . : SC65
Data blocks read . . . : 0
Data blocks written  . : 0
Dir blocks r/w . . . . : 0
Mount parameter:
   _____________________________________________
   _____________________________________________
```

A third option to display the owner is to use USS command df -v.

```
NIGELR2 @ SC64:/O39RA1/usr/lpp/dfsms/bin>df -v
Mounted on     Filesystem               Avail/Total    Files      Status
...
/u/guts        (NIGELR2.TEST.HFS)       262032/2888640 4294938692 Available
HFS, Read/Write
File System Owner : SC65
...
```

You can use the SETOMVS system command to change the setting for the AUTOMOVE parameter as shown in the next example:

```
SETOMVS FILESYS,FILESYSTEM='NIGELR2.TEST.HFS',AUTOMOVE=NO
```

```
SC64        SETOMVS FILESYS,FILESYSTEM='NIGELR2.TEST.HFS',AUTOMOVE=NO
SC64        BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.

SC64        D OMVS,F
SC64        BPXO045I 01.14.08 DISPLAY OMVS 637
            OMVS     000F ACTIVE          OMVS=(9B)
            TYPENAME   DEVICE ----------STATUS----------- MODE
            HFS            31 ACTIVE                     RDWR
            NAME=NIGELR2.TEST.HFS
            PATH=/u/guts
            OWNER=SC65     AUTOMOVE=N CLIENT=Y
            HFS            14 ACTIVE                     RDWR
```

For more information on the SETOMVS command, see *OS/390 V2R9.0 MVS System Commands*, GC28-1781, or 3.3.4, “SETOMVS system command” on page 66.

5.8 Installing service to products in the HFS

This topic has been covered in 7.3.1, "Installing Service to Products in the HFS", of the redbook *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customization,* SG24-5178.

5.9 Snapshot and DFSMSdss COPY considerations

You can use full volume copy to snap a volume containing HFS data sets. You can use DFSMSdss virtual concurrent copy (VCC) for HFS data sets during full volume, physical data set, and logical data set dump operations.

However, you cannot use Snapshot to copy an individual HFS data set by using the SNAP DATASET command. The same restriction exists for the DFSMSdss COPY function. You cannot use DFSMSdss to copy an individual HFS data set.

Snapshot serialization is similar to DFSMSdss. For a VOLUME SNAP, the enqueue is against the VTOC. See 5.1.1.6, “Physical dump” on page 112 for more detail.

For example, we receive the following messages when we try to snap an HFS data set by using the SNAP DATASET command:

```
SIB4622E  open DDN error in SIBDMOPN, rc=10.
```

As well as the following message:

```
IEC143I 213-8C,IFG0194D,STYRES1B,SNAP1,SYS00002,2557,RV2CU3,
STYRES1.TESTSNAP.HFS1
```

The message SIB4622E indicates an open error:

```
RC10 : An ABEND occurred.
```

The ABEND213 indicates that an error occurred during processing of an OPEN macro instruction for a data set on a direct access device.

```
RC8C : An OPEN macro instruction was attempted for an HFS data set, using a
DCB. BSAM/QSAM OPEN cannot be used to open a data set with DSNTYPE=HFS.
```

For a full volume copy operation, Snapshot and DFSMSdss serializes the VTOC to prevent DADSM functions (such as ALLOCATE, EXTEND, RENAME, and SCRATCH) from changing the contents of the VTOC on the volume during the copy operation. Data sets are not serialized on these full or track operations. Therefore, some data sets might be opened by other jobs during the copy, resulting in copies of partially updated data sets. You can minimize this possibility by performing the copy when there is low system activity.

A second important point, as with DFSMSdss physical dump processing, is that SNAP VOLUME does not provide quiesce capability to write cached data to disk. Your target HFS data set always reflects the HFS structure from the previous sync process. This means that newly created files can be missed and newly deleted files will still be present.

Therefore, we do not recommend SNAP VOLUME processing for HFS data sets that are currently mounted read/write.

See 5.1.1.6, “Physical dump” on page 112, and the Snapshot manuals, for more information.

The next example shows the SNAP VOLUME command. At the time this command was used, there was one HFS data set currently mounted (R/W) on the source volume.

```
SNAP VOLUME( -
   SOURCE(VOL(RV2CU3)) -
   TARGET(VOL(SMP79F)) -
   COPYVOLID(NO) -
   REPLACE(YES) -
   )
```

The job ends with RC=8 along with the following Snapshot messages:

```
SIB4627E Unable to allocate the resource RV2CU3.
SIB4617I 19:28:52 Snapshot completed, rc=8.
```

We could bypass the RC8 in one of the following ways:

- Unmount the HFS data set
- Specify TOLERATEENQFAILURE(YES) at the SNAP VOLUME command.

In spite of our example above (and examples shown in other Snapshot manuals), a SNAP VOLUME does not only serialize on the VTOC; It appears that the SNAP VOLUME processing also tries to serialize on the data set level. This processing may be changed in the future to be consistent with DFSMSdss and the Snapshot documentation.

Chapter 6. Tivoli Storage Manager

This chapter discusses the use of Tivoli Storage Manager to provide backup and recovery of HFS data at the file level. This is designed to complement, rather than replace, the backup procedure described in Chapter 5, “Managing HFS data sets” on page 103.

While you may not choose to purchase TSM solely to provide management of HFS data, (where a TSM server already exists in your organization) it is easy to install the TSM UNIX System Services (USS) client software to provide this additional support.

6.1 Introduction to Tivoli Storage Manager

Tivoli Storage Manager (TSM), previously known as ADSTAR Distributed Storage Manager or ADSM, is a client-server product designed to protect and manage a broad range of data, from Notebook PCs to powerful corporate servers. TSM supports more than 35 different operating platforms using a consistent Web-based graphical user interface (GUI).

6.1.1 Why use Tivoli Storage Manager?

TSM provides granularity to the storage management of HFS file systems. DFSMSdss DUMP provides recovery protection of entire HFS data sets, where TSM allows you to backup and restore individual files. The price paid for this level of granularity is performance. DFSMSdss DUMP works much faster than TSM, because it is not keeping track of each file, but as you will see below, you do not need to backup every file using TSM — just those data critical to your organization.

We recommend that you use *both* DFSMSdss and TSM to provide your data management. DFSMSdss DUMP will allow rapid recovery from a complete loss of HFS data sets, including system files, and TSM allows you to recover lost or damaged critical files or directories in an otherwise healthy HFS system.

Important features of TSM are:

- TSM works on a “progressive incremental” (also known as “incremental forever”) principle by default, allowing you to backup only those files which have changed since the last backup.
- TSM provides functions for long term archiving of critical files, which are tracked and protected using separate policies from the backups.
- TSM has a Central Scheduler, which allows incremental or selective backups at defined intervals. For example, you could back up all of your files nightly, and critical files in a specific subdirectory hourly.
- Separate policies can be applied to manage the data within a given HFS file system. These policies, which will be familiar to users of SMS but are not dependent on SMS, define retention periods and versioning, so that you may keep more versions of critical files and allow point-in-time restoration. These policies, together with an Include/Exclude list, allow you to control what is backed up, how often it is backed up, and how the backup data will be managed.

- According to your security requirements, you may choose to allow end users to recover files they access, or you may allow storage administrators or help desk operators to perform functions on users' behalf.

Most of the above functions are discussed in this chapter. However, it is beyond the scope of this book to provide full details of the TSM product.

For further information, see *Tivoli Storage Manager for MVS and OS/390: Quick Start,* GC35-0376. All TSM manuals are available in PDF and HTML format on the World Wide Web via URL:

`http://www.tivoli.com/tsm/`

Many redbooks have also been written about TSM. Many of them were written when the product was known as ADSM, but are still relevant to the TSM product. To view or order TSM redbooks, search for ADSM or TSM at:

`http://www.redbooks.ibm.com/`

6.1.2 Tivoli Storage Manager components

The basic components of TSM are shown in Figure 57 below. The diagram shows the following TSM building blocks:

- **Server**: The TSM server is the key component in a TSM installation. There are many supported server platforms, from Windows NT to OS/390. It is likely, but not imperative, that you will use an OS/390 server to manage your HFS data. The managed client data is tracked by an internal relational database. The server also contains a Central Scheduler to control the frequency of client file capture.
- **Storage Pools**: This is where the TSM server stores backup or archive objects on behalf of the managed clients. This may be automatically managed as a hierarchy of disk, optical, or tape storage.
- **Clients**: For the purposes of this book, we will only be describing the OS/390 UNIX System Services client. TSM clients are available for many other operating systems, providing data management for the whole of your enterprise. There are two types of TSM clients:
- **Backup/Archive**: This client is used to backup, restore, archive and retrieve data on behalf of the managed machine. Command line and GUI (via Web interface) clients are available.
- **Administration**: This a special client for managing TSM servers. Command line and GUI (via Web interface) clients are available. TSM servers running under OS/390 can also be managed via operator console *modify* commands.

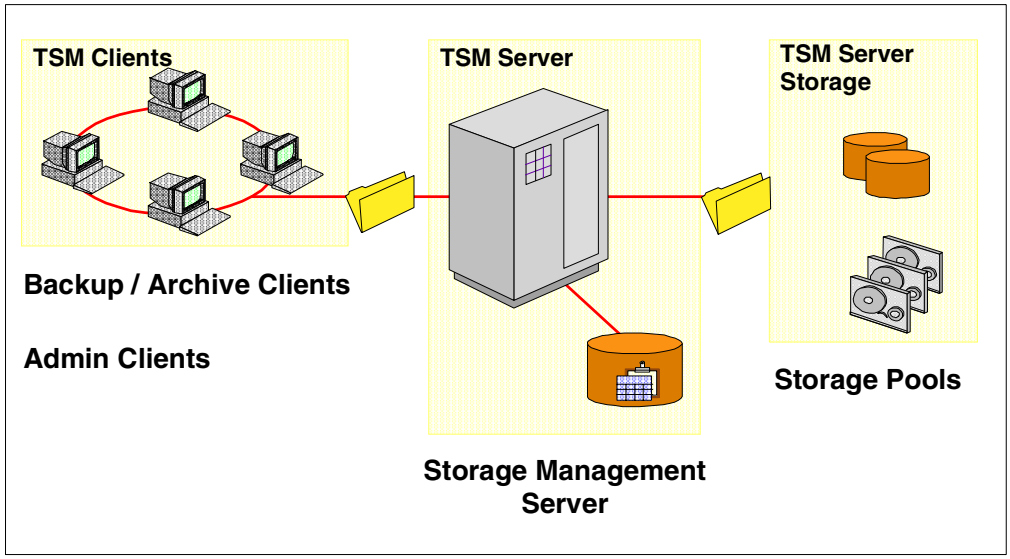


Figure 57. Tivoli Storage Manager components

6.2 Installing the Tivoli Storage Manager USS client

This section describes the basic steps for setting up the TSM client for USS. This information should be used in conjunction with the manuals referenced below.

6.2.1 Ordering the client software

Most TSM client software is freely downloadable from the World Wide Web, via `http://www.tivoli.com/tsm/`. This is not the case with the USS client. The USS client is delivered with the TSM for OS/390 server product, or may be ordered as a service tape from your Customer Support organization.

6.2.2 Initial installation

This section describes the basic steps needed to install the USS client software. These instructions should be read in conjunction with the *Program Directory for the Tivoli Storage Manager Backup-Archive,* GI10-4520.

1. Perform `SMP/E RECEIVE` for the product tape. Sample JCL is provided in the Program Directory.
2. Sample JCL is provided for `pre-APPLY, APPLY-CHECK and APPLY` jobs. Customize these jobs according to your organization defaults and submit. The jobs should all complete with rc=0.
3. Run the `Post-APPLY` steps described in the Program Directory. You are now ready to customize the client options files.

6.2.3 Customizing the client

See *Tivoli Storage Manager for UNIX: Using the Backup-Archive Clients,* SH26-4105, for full instructions on customizing the client options files. Following the basic instructions in this section should leave you with a working USS client. There are many additional options for customizing TSM, outside the scope of this book.

Note

The instructions in the TSM manuals for copying the sample options files to files in the installation directory conflict with the recommended mount mode for the Version HFS root file system in OS/390 USS. IBM recommends that the Version root file system should be mounted read-only (see 10.2.5.2, “VERSION” on page 239). In order to customize these files, you must copy them to a read-write file system. We have chosen to use `/var/tsm` to hold the client options files in our system

We copied the sample files to `/var/tsm`., a file system mounted R/W. You may also need to copy the language file `dscameng.txt` from the installation directory to your chosen R/W file system, to avoid errors starting the TSM client. The sample screens below show the `TSO OMVS` environment. You may choose to use `telnet` or `X-Windows` to run similar commands:

Now you need to copy the sample client options files `dsm.opt.smp` to `dsm.opt` and `dsm.sys.smp` to `dsm.sys`, as shown in the `OMVS` example below, then edit them to provide connectivity to your TSM server.

```
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 2000
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

NIGELR3 @ SC64:/>cd /var/tsm
NIGELR3 @ SC64:/SC64/var/tsm>cp dsm.opt.smp dsm.opt
NIGELR3 @ SC64:/SC64/var/tsm>cp dsm.sys.smp dsm.sys
NIGELR3 @ SC64:/SC64/var/tsm>

 ===>
                                                                    RUNNING
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

Examples of basic `dsm.opt` and `dsm.sys` and Include/Exclude files are shown in Figure 58, Figure 59, and Figure 60 below. **Note**: To save space, the Include/Exclude list shown in Figure 60 is not a complete copy of the list used in our environment.

```
SErvername        WTSCMXA
```

Figure 58. Example dsm.opt file

```
SErvername       WTSCMXA
COMMmethod         TCPip
TCPPort            1500
TCPServeraddress   wtscmxa.itso.ibm.com
PasswordAccess     Generate  1
PasswordDir        /var/tsm
NODename           WTSC64OE
SCHEDLOGName          /var/tsm/dsmsched.log
SCHEDLOGRETENTION 14 D                      2
ERRORLOGName          /var/tsm/dsmerror.log
ERRORLOGRETENTION 14 D                      2
INCLexcl           //'NIGELR3.TSM.PARMLIB(WTSC64OE)' 3
```

Figure 59. Example dsm.sys file

1 `PasswordAccess Generate` saves an encrypted copy of the client password in the directory specified by `PasswordDir`.

2 It is worth automatically deleting log records, but you should regularly monitor these files! Here we specify 14 days.

3 You may use a PDS or PDSE member for the Include/Exclude list. Note the special format of the file name needed to achieve this.

Note the special syntax in Figure 60. As well as asterisk (*) wildcards which are familiar to most users, we also have `/.../` (known as the “match-n” wildcard), which refers to any number (0-n) of subdirectories. Full syntax rules are described in *Tivoli Storage Manager for UNIX: Using the Backup-Archive Clients,* SH26-4105.

```
EXCLUDE /.../*
EXCLUDE.DIR /bin
EXCLUDE.DIR /bin/.../*
EXCLUDE.DIR /dev
EXCLUDE.DIR /dev/.../*
EXCLUDE.DIR /samples
EXCLUDE.DIR /samples/.../*
EXCLUDE.DIR /tmp
EXCLUDE.DIR /tmp/.../*
EXCLUDE.DIR /var/adsm/logs
EXCLUDE.DIR /var/adsm/logs/.../*
*
INCLUDE /usr/local/.../*
INCLUDE /etc/.../*
INCLUDE /u/.../*
INCLUDE /var/.../*
```

Figure 60. Example Include/Exclude list

The basic setup is now complete. You have told the client where to find a TSM server, and limited the files being backed up to critical data in your organization. In order to point the TSM client at these options files, we need to set three environment variables:

- **DSM_DIR**: Points to the `dsm.sys` file
- **DSM_CONFIG**: Points to `dsm.opt` file
- **DSM_LOG**: Points to the directory where the `dsmerror.log` file will be written

In the Figure 61 below, we are setting all these environmental variables to point to our R/W directory `/var/tsm`:

```
NIGELR3 @ SC64:/SC64/var/tsm>cd /$HOME
NIGELR3 @ SC64:/>DSM_DIR=/var/tsm
NIGELR3 @ SC64:/>DSM_CONFIG=/var/tsm/dsm.opt
NIGELR3 @ SC64:/>DSM_LOG=/var/tsm
NIGELR3 @ SC64:/>export DSM_DIR DSM_CONFIG DSM_LOG
NIGELR3 @ SC64:/>

===>
                                                                        INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

Figure 61. Setting environment variables

Adding the above environment variables to `/etc/profile` will set them each time a USS session is started.

Registering the client

The TSM server needs to know that your client is authorized to backup, archive, restore, or retrieve files. To complete the client installation, you need to register your client node name on the server. This needs to be performed by a TSM administrator. If you are responsible for both the client and server, you will perform this task. Otherwise, see your storage administrator!

6.2.4 Using the Tivoli Storage Manager client

You are now ready to test the TSM USS client. The following sections describe the basic operation of the client software. You should consult *Tivoli Storage Manager for UNIX: Using the Backup-Archive Clients,* SH26-4105, for more detailed instructions.

TSM provides two forms of interface to the client software:

- **Command line**: This interface is provided so that “dumb” terminals, such as telnet or OMVS can be used to access the client. Command line does not look as friendly, but can perform all TSM functions.
- **GUI**: This interface is provided via a Java applet on a suitable Web browser. This not only makes the interface easier to use for ad-hoc file restoration, but it can be used on any machine that has a Web browser, anywhere that can get TCP/IP access to your client. It is particularly useful for help-desk personnel to restore lost files on your behalf.

6.2.4.1 Command Line interface

The command `dsmc i` (incremental backup) should produce similar output to the screen shown in Figure 62 (actual output will depend on your system environment).

```
Tivoli Storage Manager
Command Line Backup Client Interface - Version 3, Release 7, Level 2.0
(C) Copyright IBM Corporation, 1990, 2000, All Rights Reserved.
.
. (output lines removed for clarity)
.
Selective Backup processing of '/u/tsmnew' finished without failure.

Total number of objects inspected:   56,283
Total number of objects backed up:       80
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:          1
Total number of objects expired:          0
Total number of objects failed:           0
Total number of bytes transferred:   511.93 KB
Data transfer time:                    0.32 sec
Network data transfer rate:        1,566.95 KB/sec
Aggregate data transfer rate:         10.74 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:00:47
```

Figure 62. Typical TSM command line interface output

Commonly used commands (showing short command format):

i perform incremental backup

sel perform selective backup (backup individual file(s) with wildcards), for example, `sel /u/tsmnew/*.tar`

ar archive file(s), with similar syntax to `sel`

res restore file(s); includes a picking list function (Figure 63 below)

```
TSM Scrollable PICK Window - Restore

     #    Backup Date/Time            File Size A/I  File
        ---------------------------------------------------------------------
    76. | 03/18/2000 16:57:06     244.00 KB  A   /u/local/tsmtest/acl_edit
    77. | 03/18/2000 16:57:06      68.00 KB  A   /u/local/tsmtest/alias
x   78. | 03/18/2000 16:57:06     176.00 KB  A   /u/local/tsmtest/ar
x   79. | 03/18/2000 16:57:06      76.00 KB  A   /u/local/tsmtest/asa
x   80. | 03/18/2000 16:57:06     184.00 KB  A   /u/local/tsmtest/at
    81. | 03/18/2000 16:57:06     364.00 KB  A   /u/local/tsmtest/awk
x   82. | 03/18/2000 16:57:09       3.30 MB  A   /u/local/tsmtest/bak
x   83. | 03/18/2000 16:57:12      60.00 KB  A   /u/local/tsmtest/basename
x   84. | 03/18/2000 16:57:12     184.00 KB  A   /u/local/tsmtest/batch
x   85. | 03/18/2000 16:57:12     240.00 KB  A   /u/local/tsmtest/bc
    86. | 03/18/2000 16:57:12      68.00 KB  A   /u/local/tsmtest/bg
    87. | 03/18/2000 16:57:14       1.64 MB  A   /u/local/tsmtest/bos
    88. | 03/18/2000 16:57:14       2.01 KB  A   /u/local/tsmtest/bpxmtext
    89. | 03/18/2000 16:57:18     488.00 KB  A   /u/local/tsmtest/c++
    90. | 03/18/2000 16:57:27     488.00 KB  A   /u/local/tsmtest/c89
        0---------10--------20--------30--------40--------50--------60--------7
<U>=Up  <D>=Down  <T>=Top  <B>=Bottom  <R#>=Right  <L#>=Left
<G#>=Goto Line #  <#>=Toggle Entry  <+>=Select All  <->=Deselect All
<#:#+>=Select A Range <#:#->=Deselect A Range  <O>=Ok  <C>=Cancel
pick>
```

Figure 63. The TSM PICK Window

The “Scrollable PICK Window” may look a little messy, but is quick and easy to operate, once you get used to it! In Figure 63 we have selected a range of files for restore, shown by the `x` characters on the left of the screen.

6.2.4.2 GUI interface

From version 3.7.2, TSM provides a Web GUI interface for accessing functions via a convenient Web browser interface. The interface, which is provided by means of Java applets, is of particular use by help desk personnel. From Web browsers anywhere in your organization, you can access any client's backup files or administer your TSM server(s), depending on the security access given to the user.

To activate TSM's Web GUI interface, you need to know the URL of the TSM Web client. The URL is made up of the client's IP address and the port number on which the Web admin client is listening. The default Web client port address is 1581. For our system, the URL becomes `http://wtsc64oe.itso.ibm.com:1581/`

An example of the front page displayed by the Java GUI is shown in the example using Netscape Communicator in Figure 64. The four primary functions are provided by the first large buttons:

- **Backup**: Perform incremental or selective backup of the HFS file space
- **Restore**: Restore files to original or new location
- **Archive**: Archive files for long term storage
- **Retrieve**: Retrieve archived files

The last button, **Getting Started**, provides an overview of TSM functions.

We only deal with the main TSM functions in this book. Please see the TSM user manuals (discussed in 6.1.1, "Why use Tivoli Storage Manager?" on page 153 and page 156) for further information about the client interfaces.

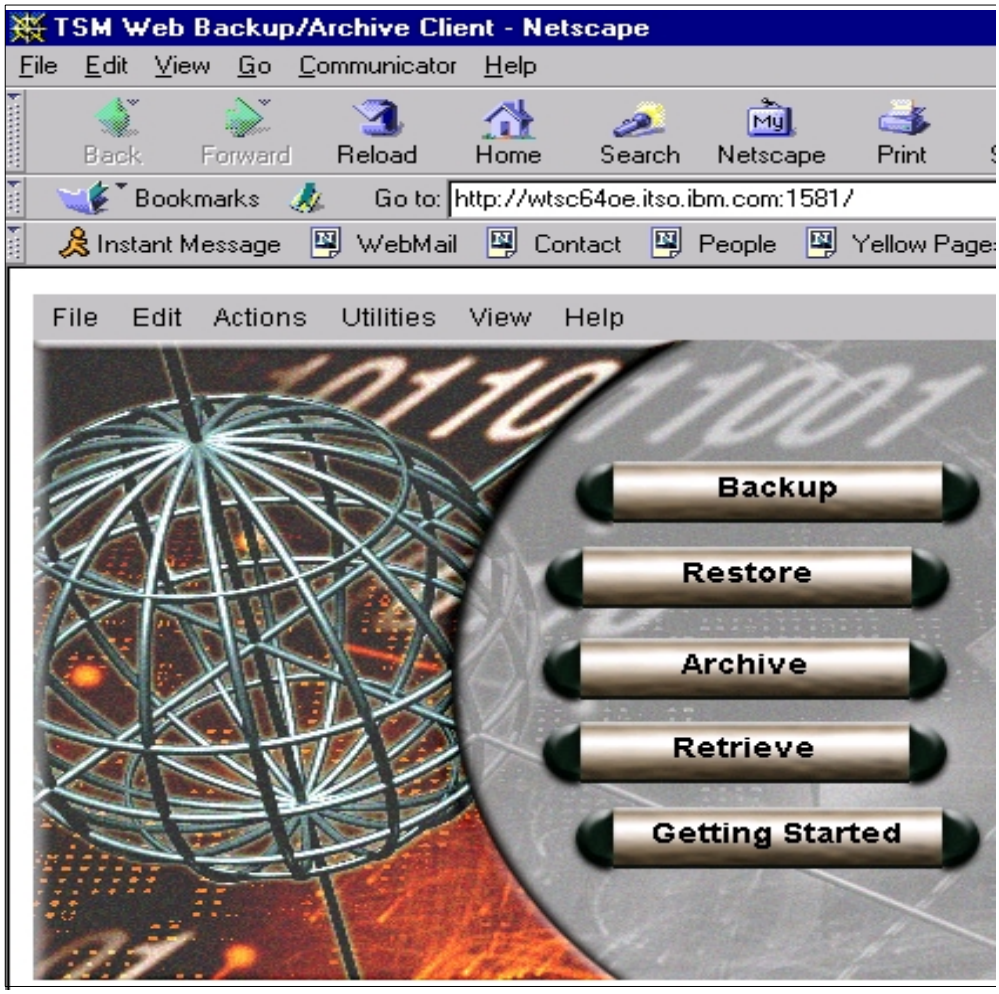


Figure 64. The TSM GUI front end

Pressing any of the main function buttons (Backup, Restore, Archive, Retrieve), will result in a request for a valid User ID and Password, as shown in Figure 65. The authentication is performed at the TSM **server**, so a valid administrator ID and password must be entered. There is a special **Client Owner** ID available. This ID cannot administer the server, but is allowed to activate client functions through the Web interface. You may visit the Getting Started overview without logging in.

Figure 65. TSM Login panel

On successful login, TSM will generate a new Java window showing a tree view of your file system, similar to that shown in Figure 66 (this figure shows the Backup window — the tree view and its actions are similar for all four functions).

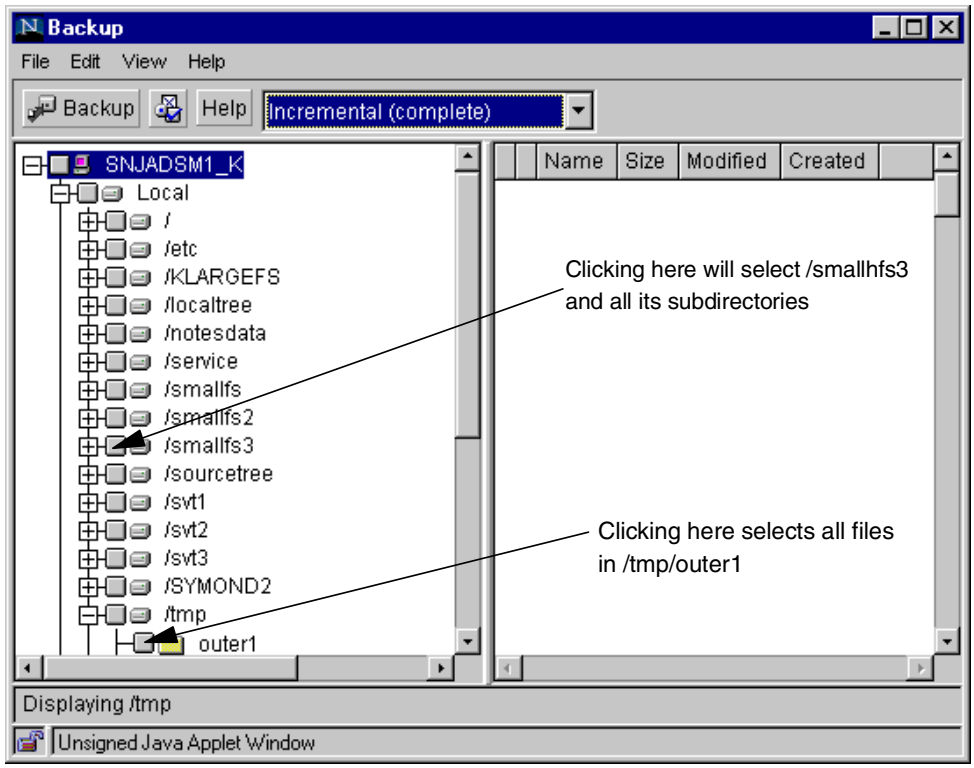


Figure 66. TSM Backup - tree view

From this tree view, you may select complete subdirectories or individual files. Examples are shown by arrowed comments.

Note: Files excluded in the Include/Exclude list (Figure 60 on page 157) are not selectable, and are shown by a red x on the tree view. If you need to back up such files, you must edit the Include/Exclude list.

In this section, we have provided a short tutorial in TSM major functions. It is outside the scope of this book to provide a complete functional guide, but hopefully has been enough to get you started!

6.2.5 TSM server considerations

Server placement is dependent on many factors. The most probable scenario for the USS client is that you will have an existing server in your organization. If you are starting with a clean sheet, the best performer is likely to be TSM for OS/390 server running on the same machine or Sysplex as the USS client. Alternatively, you could consider using an Open Systems Adaptor (OSA) to connect to TSM servers elsewhere in your organization. Such is the flexibility provided by TSM, you may even back up your USS environment to a TSM server running under Windows NT, although we are unaware of any customers taking this approach!

6.2.6 HFS file sharing considerations

Because the TSM client node name defaults to the hostname, if you need TSM to restore a shared file system from multiple nodes you would need a backup to be performed from every sysplex node, resulting in the same files being backed up more than once. This is, of course, very wasteful of server storage. One way around this is to specify the same NODename (see Figure 59 on page 157) for *every* client which may be backing up or restoring the shared file system.

Now, the TSM server will be aware if a backup exists for any part of the shared file system, regardless of the node performing the backup. Note that only one system needs to perform the backup, since TSM operates on a file tree. The tree for a shared file system will look identical throughout the Sysplex. There are a number of non-shared file systems or directories (/etc, /tmp,...) unique to each Sysplex host, but these are normally excluded from TSM processing.

Note: If you feel that you *do* need to run TSM from multiple systems, you will need to apply the same changes to dsm.sys, etc., in the read-write file system for each of the Sysplex nodes that will using TSM. Consider these scenarios:

1. To run TSM from multiple systems:
   - Copy dsm.sys and dsm.opt to each system. This will allow changes to be made, if desired.
   - Create a symbolic link to the dsm.sys directory in the “owning” system (this is preferred, because it forces the `NODename` to be identical on all systems, assuming you have specified this parameter in dsm.sys).
2. To run TSM from one system:
   - The Include/Exclude list should reflect all systems. For example:

     ```
     exclude /SC64/var/.../*
     exclude /SC65/var/.../*
     .
     .
     ```

     Where SC64 and SC65 are the names of individual systems.

6.3 Data management strategies

This section discusses the various options available for managing your data with the aid of TSM.

6.3.1 DFSMSdss and Tivoli Storage Manager

As we discussed earlier in this chapter, TSM is not designed to replace your normal dump/restore processing. Both methodologies clearly have a part to play. In the same way that you might use mksysb to create a system image to rapidly restore an AIX system, DFDSMdss dump/restore provides a similar function in an OS/390 USS environment. In both scenarios, you are using a rapid facility to save you system image, while TSM provides data management at file level and online backup and restoration of lost or damaged user data.

You would not normally use TSM to backup system files, since these do not change frequently, although you are not precluded from this if your storage strategy demands it. TSM’s “progressive incremental” approach means that the

overhead of backing up such files occurs only on the first run. Thereafter, only changed files will be considered for backup. However, it is because of the reduced restore performance that it is beneficial to take regular dumps of HFS volumes and use TSM to recreate files which have changed and been backed up since.

6.3.2 Disaster recovery considerations

Your recovery methodology will depend upon the degree of failure. Accidental loss or damage of critical files may be recovered merely by using the TSM USS client. This may be achieved while HFS is active, giving access to other data during the restore process. Disaster recovery assumes a catastrophic failure, for instance loss of a complete DASD volume. In such cases, you will need to invoke the services of DFSMS as well as TSM to recover your lost system.

Assuming a total loss of a volume containing HFS data, the following recovery process may be used to bring your HFS files to the time of the TSM backup:

- Recover the volume using DFSMShsm

  Check for existence of multi-volume HFS data sets. Recover these with HSM:

  ```
  RECOVER NIGELR2.HFS.DATA GEN(0)
  ```

- Use TSM to apply updates since last DUMP:

  ```
  dsmc res * -sub=yes -replace=yes -ifnewer
  ```

  Note: There may have been files and/or directories added since the last TSM backup. It may be difficult to recover from this situation, or even discover that such changes have been made!

In B.6, “Recovery samples” on page 284, it shows a more complex disaster recovery example, where TSM is being used along with system tools, to recover from a broken file system. In the example, we have deliberately made part of an HFS unavailable by zapping parts of the index.

6.4 Performance considerations

This section discusses some ways to get the best performance out of TSM. Performance depends on many factors, including network load and server placement. Most of these factors may be outside your immediate control. Here we discuss parameters at your disposal which will affect the data management performance of your HFS.

6.4.1 Tuning

TSM provides a number of tuning parameters for both client and server components. These parameters are set by editing dsm.sys (described in Figure 59 on page 157). If the parameters are omitted, a default will be used.

Here we show some of the available parameters, with their recommended and default values. The capitalized letters show the minimum characters required to be typed into dsm.sys:

- **COMPression No**: Default — **No** (compaction by the tape drive is more efficient). Compression is performed at the client, so may be beneficial if the network bandwidth to the TSM server is very low).

- **Quiet**: Default — **VERBOSE** (verbose messages are shown in this chapter. Quiet turns off messages to screen and reduces I/O to the log files. Error messages and summary information still appear in the logs).
- **TCPBuffsize 512**: Default — **8** (size of the TCP buffer, in kilobytes) **Note**: This parameter is also valid on the TSM server.
- **TCPWindowsize 640**: Default — **32** (size of the TCP sliding window, in kilobytes. A larger window size can improve communication performance, but uses more memory. The maximum value is 2048) **Note**: This parameter is also valid on the TSM server.

Two command line options exist, which will also influence the performance of TSM. Command line options are preceded with a dash (-). Examples of IFNEWER processing are shown in B.6, “Recovery samples” on page 284:

- **IFNEWER**: This option is used in conjunction with the restore command, and restores files only if the server date is newer than the date of the local file. This option will affect network utilization because less data needs to travel across the network.
- **INCRBYDATE**: In a normal incremental backup, the server reads the attributes of all the files in the file system and passes this information to the client. The client then compares the server list to a list of its current file system, backing up files which have changed. For an “incremental by date” backup, the server only passes the date of the last successful backup, avoiding the need to query every active file on the ADSM server. The time savings are significant. However, periodic regular incremental backups are still needed to backup files that have only had their attributes changes. For example: If a new file in your file system has a creation date earlier than the date of the last successful backup, any future INCRBYDATE processing will not backup this file, because the client will believe that it has already been backed up.

Finally, use the USS client’s Include/Exclude list (Figure 60 on page 157) to eliminate unnecessary backups. Excluded files will still be protected by your normal dump/restore processing.

6.4.2 Reorganizing HFS for performance

Table 6 shows the TSM performance improvements gained at a customer site by reorganizing the HFS data, using the copytree utility (described in B.6, “Recovery samples” on page 284).

The table shows data rates and elapsed time for TSM backup processing before and after using copytree to defragment the HFS data.

The key point when viewing this table is not the raw data, but the fact that an average performance improvement of more than 47% has been achieved by reorganizing the HFS to reduce fragmentation.

Table 6. HFS performance improvement following copytree reorganization

Before	After	Number of files transferred to TSM	Data transferred in GB	Duration in minutes	
in GB / hour				Before	After
10.1	14.8	1085	36	210	147
9.5	13.0	1783	23	144	107
5.5	11.3	1788	23	255	126
12.0	18.4	1043	37	186	121
6.6	13.4	1086	39	353	175
15.0	18.3	417	26	103	85
11.5	16.3	418	22	117	81
10.3	13.5	416	21	120	93

Note

TSM is unlikely to be the only software to gain from the data being in a defragmented state, so regular HFS reorganization is to be recommended. See 5.3.1.1, “Copying an HFS” on page 127 for further information on the defragmentation process.

Chapter 7. Sharing and serialization for HFS data sets

This chapter provides information about sharing and serialization for HFS data sets.

We distinguish between two different kinds of sharing:

- Sharing an HFS data set in read-only mode.
- Sharing an HFS data set in read/write mode.

Figure 67 shows what happens when you mount an HFS data set in **read-only** mode on multiple systems. This kind of sharing is independent of the levels of the individual systems. For example, system 1 can be an OS/390 2.7 system, system 2 can be at level OS/390 2.8, and system 3 is an OS/390 2.9.

Note

Sharing an HFS data set in read-only mode is like sharing DASD. Every system reads HFS metadata and data directly from DASD.

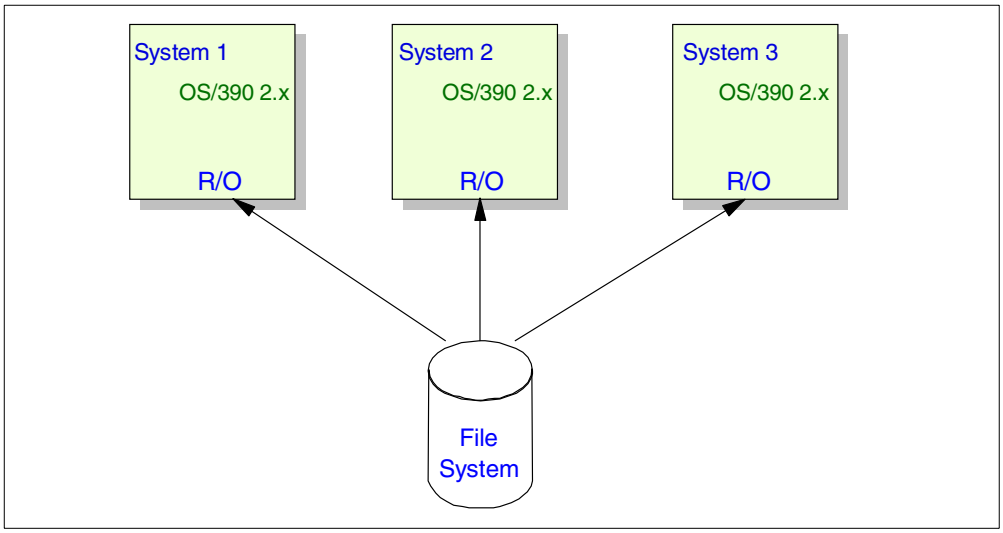


Figure 67. Sharing a file system in read-only mode

Starting with OS/390 2.9, you can share an HFS data set in **read/write** mode between other OS/390 2.9 systems (or above) in a *participating group* (Figure 68). Here are several points:

- The term *participating group* identifies those systems that belong to the same SYSBPX XCF sysplex group.
- To be in the participating group, the OS/390 system level must be 2.9 or later with DFSMS/MVS 1.5
- You need to specify SYSPLEX(YES) in your SYS1.PARMLIB(BPXPRMxx) member

See 7.3, “HFS sysplex sharing (OS/390 2.9 and above)” on page 179 regarding information about HFS sysplex sharing.

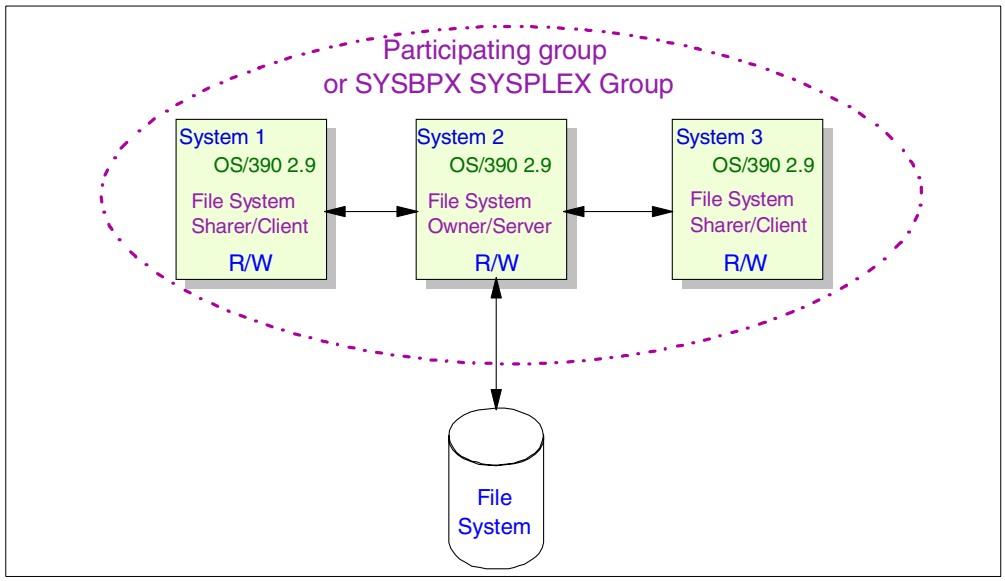


Figure 68. Sharing a file system in read/write mode

Note

HFS Sysplex sharing is implemented by *function shipping* and not by using shared DASD.

As shown in Figure 68, HFS sysplex sharing is implemented as a kind of *function shipping* and not by using shared DASD. This means that one system owns a file system (the server system) and it serves the other sharing systems (clients). Only the owning system performs physical I/Os to read and write the data and metadata from an HFS data set. The client systems send the requests to the server system for the particular HFS. The clients do not access the HFS data set directly.

From the DFSMS/MVS HFS point of view, the same serialization rules apply to HFS sharing in a sysplex as existed before OS/390 2.9, because only the server system performs the physical access to the HFS data set on DASD in a participating group. This means that you cannot share an HFS data set outside a participating group if you have mounted the file system read/write in a participating group. For more information, see:

- Section 7.1, “Serialization considerations” on page 170
- Section 7.3, “HFS sysplex sharing (OS/390 2.9 and above)” on page 179

Therefore, it is not possible to share an HFS data set in read/write mode that is currently mounted on an OS/390 2.9 system with OS/390 2.8 (or below) systems (see Figure 69). If you want to share the HFS data set, it must be mounted in read-only mode on all systems. The same restriction exists when you have mounted an HFS data set in read/write mode on an 2.8 (or below) system. You cannot mount the same HFS on any 2.9 system to share its read/write. See 7.2, “Sharing considerations (OS/390 2.8 and below)” on page 174 for more details regarding sharing an HFS data set with OS/390 2.8 (and below) systems.

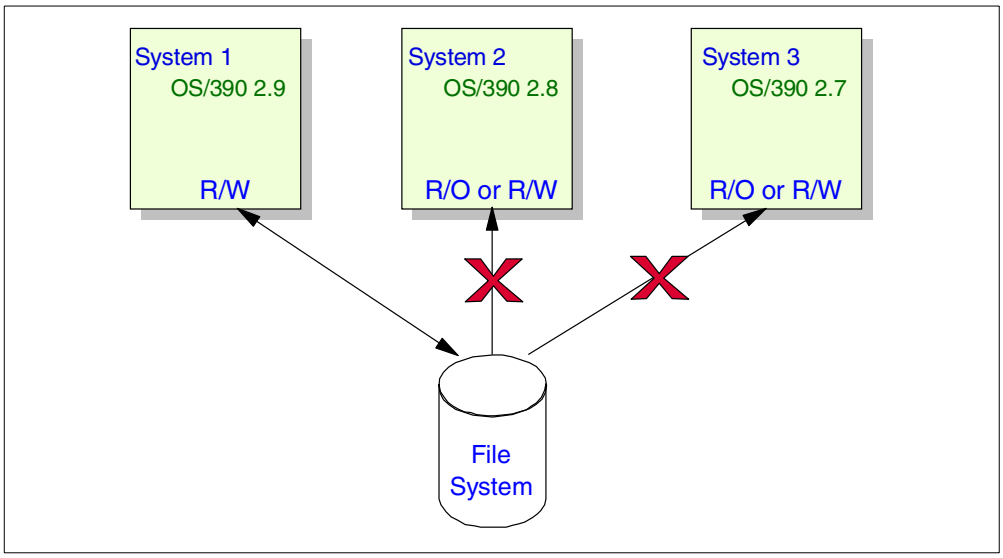


Figure 69. Sharing restriction with systems before OS/390 2.9

It is also not possible to share an HFS data set in read/write mode with other systems outside the participating group (see Figure 70).

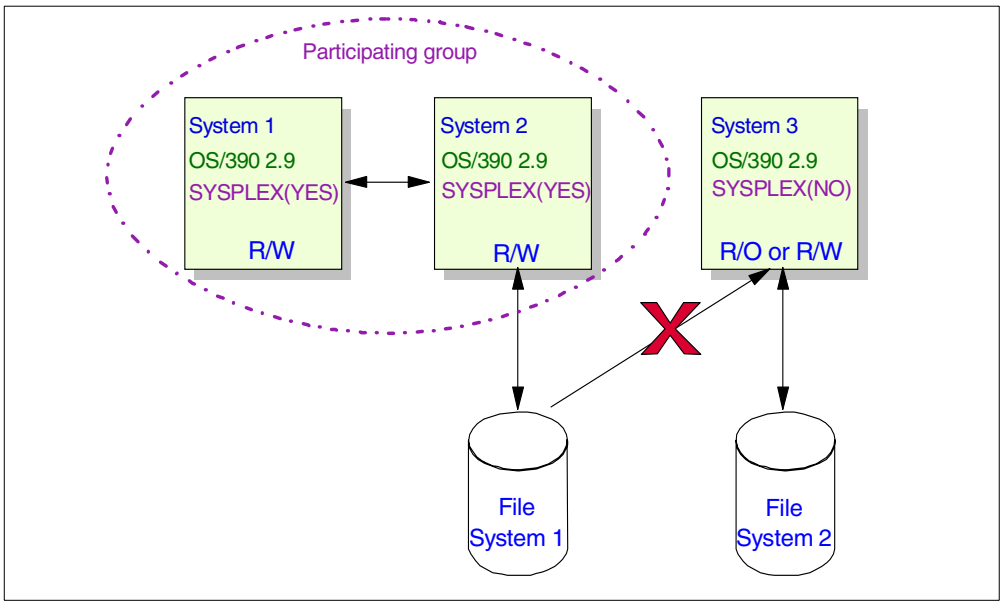


Figure 70. Sharing restriction for systems outside the participating group

If you want to share an HFS data set between systems outside a participating group, then you must mount the HFS data set in read-only mode on all systems as shown in Figure 71.

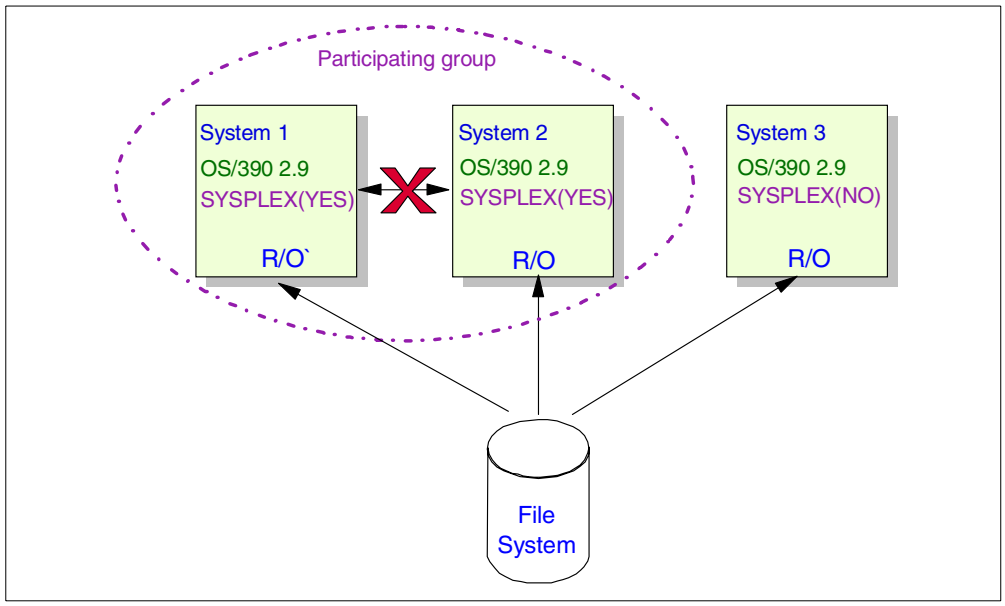


Figure 71. Sharing an HFS in read-only mode in a sysplex

The benefit of this kind of implementation is that the data and metadata don't need to be shipped between the systems in a participating group and therefore you will get better performance. However, you lose the possibility to mount the file system in read/write mode on any of these three system without first doing an unmount on all three systems. Otherwise, you violate the sharing rules as illustrated in Figure 69 and Figure 70.

Section 9.3, "UNIX System Services (OS/390 2.9 and later)" on page 212 summarizes the benefits and drawbacks of mounting an HFS as read/write or read-only. Also in this section, OS/390 2.9 introduces a new file system structure.

Chapter 10, "HFS sysplex sharing implementation" on page 221 contains further information about the implementation of HFS sysplex sharing.

7.1 Serialization considerations

The main serialization used by HFS mount processing is a SYSTEMS ENQ on major name (qname) of **SYSZDSN** and minor name (rname) of the HFS data set name while an HFS is mounted. An ENQ on qname of **SYSDSN** and rname of the HFS data set name will also be obtained.

Table 7 provides an overview of the ENQ usage for HFS data sets that are mounted read/write or read-only.

Table 7. HFS serialization - enqueue usage

	SYSZDSN ENQ	**SYSDSN ENQ**
HFS is mounted R/W	Exclusive	Shared
HFS is mounted R/O	Shared	Shared

The SYSZDSN and the SYSDSN ENQ are held on the HFS until the file system is unmounted.

Note: In DFSMS/MVS 1.4 and older systems, if the SYSZDSN ENQ is not properly maintained the HFS data set will be damaged. Starting with DFSMS 1.5, due to the new *write protection* function the HFS index should remain consistent. However, the files within the HFS data set (the data files) could be inconsistent. See 7.2, “Sharing considerations (OS/390 2.8 and below)” on page 174 for additional information.

APARs OW30738 and OW30744 have changed the ENQ on HFS file systems from RNL=NO to RNL=YES. The fixes (PTFs) for both APARs are already included in the base of DFSMS/MVS 1.5.

The ENQ of SYSZDSN is issued with the scope SYSTEMS and RNL=YES, and therefore should NOT be placed in member GRSRNLxx when using GRS. Other serialization products may need to be customized to ensure that SYSZDSN is propagated to all systems in the complex.

APAR OW30729 provides more information on the usage of RNL=YES and NO.

Note: This documentation APAR is intended to clarify the proper use of the ENQ RNL= specification so that IBM and vendor products that either are already using, or are considering using, the enqueue specification understand its impact.

For example, you will receive the following message at mount time if you try to mount the same HFS on a second system in the same sysplex:

```
BPXF135E RETURN CODE 00000072, REASON CODE 5B220117.  THE MOUNT FAILED FOR
FILESYSTEM hfs.data.set.name.
```

- The return code `x'0072'` (EBUSY) indicates that the resource (the HFS data set) is busy.
- The reason code `x'0117'` indicates that the ENQ for qname SYSZDSN has failed.

Also, if you try to mount the HFS that is already mounted read/write on the same system, then you will receive message:

```
BPXF135E RETURN CODE 00000079, REASON CODE 055B005B.  THE MOUNT FAILED FOR FILE
SYSTEM hfs.data.set.name.
```

- The return code `x'0079'` (EINVAL) indicates an incorrect parameter.
- The reason code `x'005B'` (JRIsMounted) indicates that the file system is already mounted.

One situation that resulted in a duplicate mount in the past happened when building or cloning a new or test system based on a production system. You had to ensure that PARMLIB member BPXPRMxx in the cloned system was altered so that it did not contain the same HFS name(s) as found in the production system's BPXPRMxx member. Failing to do this would damage the HFS that was being shared accidentally, even if it was just for a few minutes after the first IPL of the new test system.

7.1.1 ENQ usage for shared HFS (OS/390 2.9 and higher)

Figure 72 shows that only one system holds the SYSZDSN ENQ in a shared HFS environment if the file system is mounted in read/write mode.

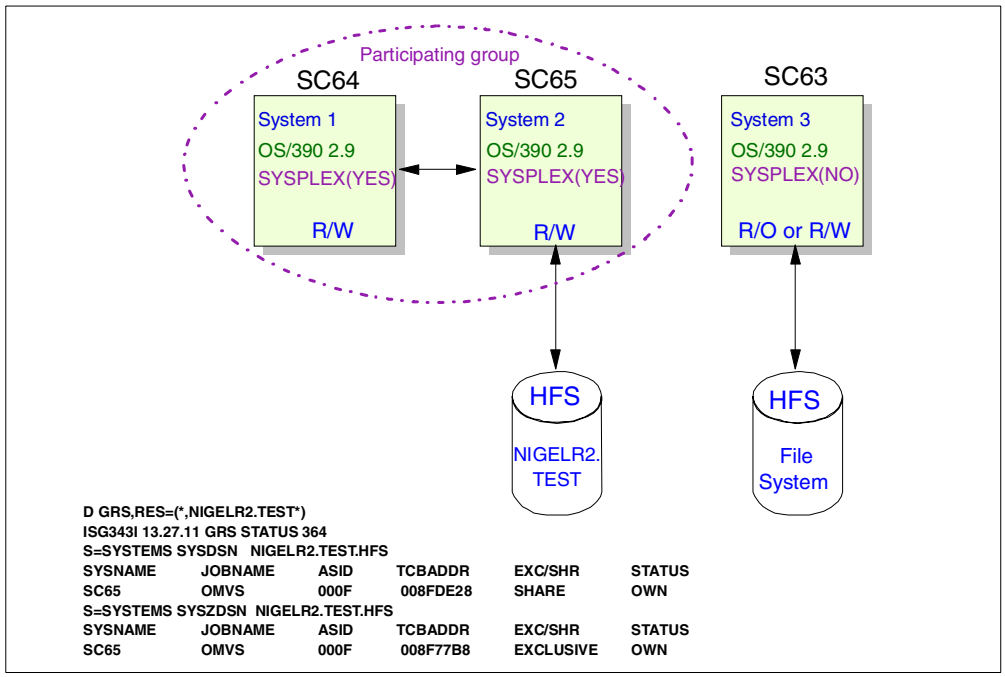


Figure 72. Enqueues for shared HFS in read/write mode

However, if you mount the file system in read-only mode, then each system in the participating group holds a SYSZDSN ENQ for the particular file system (as shown in Figure 73).

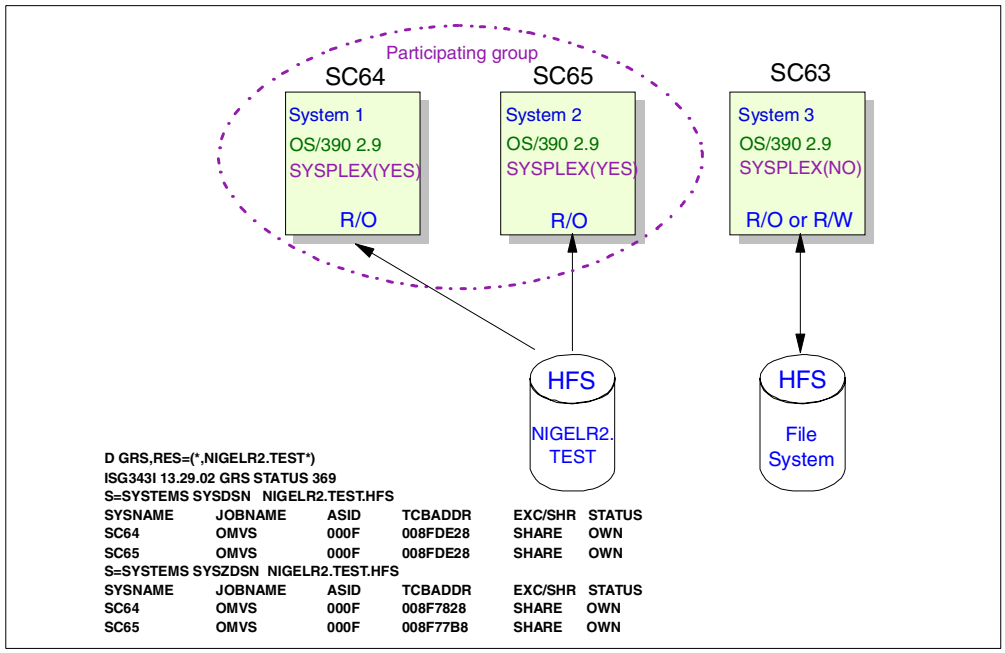


Figure 73. Enqueues for shared HFS in read-only mode

This is because, in read/write mode, the ENQ for SYSZDSN will be obtained for the physical file system (PFS) in the owning system. In read-only mode, the PFS

in each system has access to the file system. See 7.3.1, “Function shipping” on page 179 for more information.

7.1.2 Serialization by DFSMSdss

See 5.1, “DFSMSdss dump and restore” on page 103 for additional information regarding DFSMSdss DUMP processing and recommendations about dumping an HFS.

7.1.2.1 Logical dump

Starting with OS/390 2.9, you can perform a logical data set dump from any of the systems in a participating group. The DFSMSdss serialization processing is the same as in the environment before OS/390 2.9.

The serialization mechanism during logical dumps of HFS data sets consists primarily of the SYSZDSN ENQ and use of the quiesce (BPX1QSE) service.

If a data set is not mounted while it is being dumped, then serialization is provided by a shared SYSZDSN ENQ. In order to mount an HFS data set for update, OMVS must obtain an exclusive ENQ on the SYSZDSN. Thus, the shared ENQ on the SYSZDSN prevents the data set from being mounted during the dump.

If an HFS data set is mounted prior to a DFSMSdss dump job, the data set will be quiesced as long as the DFSMSdss dump job is executed on the same system that the data set is currently mounted on. The quiesce prevents updates to the data set for the duration of the dump.

In DFSMSdss 1.4 and older releases, DFSMSdss also obtains a SYSDSN ENQ. If the SHARE keyword is not specified, then DFSMSdss will attempt the SYSDSN ENQ as exclusive. If a data set is currently mounted, then OMVS will already have a shared SYSDSN ENQ. So, SHARE is required when dumping mounted HFS data sets on systems prior to DFSMSdss 1.5.

In DFSMSdss 1.5 the SYSDSN ENQ is no longer obtained by DFSMSdss for HFS data sets, so the SHARE keyword is no longer required to dump mounted HFS data sets.

Note

For shared HFS in OS/390 2.9 and higher:

During our tests, from time to time, we received a DFSMSdss message `'ADR412E ... FAILED SERIALIZATION'` during logical dump processing. This was because our DFSMSdss DUMP job was routed to a system which was not part of the participating group.

Please keep in mind, that you can specify `/*JOBPARM SYSF=systemname` in your job to route the job to a specific system which is part of the SYSBPX group.

For example (see Figure 72 on page 172), we submitted the job on system SC64, but the job was routed to system SC63. System SC63 does not belong to the participating group. Therefore we received message ADR412E correctly.

7.1.2.2 Physical dump

DFSMSdss relies on a SYSDSN ENQ for physical data set dump operations.

By default, DFSMSdss will attempt an exclusive SYSDSN ENQ during a physical dump. As long as the data set is not mounted at dump time, the exclusive enqueue on the SYSDSN resource provides sufficient serialization.

If the data set is already mounted before the physical dump job begins, OMVS will have a shared ENQ on the SYSDSN. DFSMSdss will therefore not be able to obtain an exclusive SYSDSN ENQ.

Note: with APAR OW39883, DFSMSdss has been changed to get an exclusive SYSDSN ENQ during physical data set DUMP of PDSE or HFS data sets, unless the SHARE keyword is specified.

For full volume processing (physical), DFSMSdss issues a RESERVE command on SYSVTOC (this is the default).

```
D GRS,RES=(SYSVTOC,*)
ISG343I 12.50.52 GRS STATUS 977
S=SYSTEMS SYSVTOC  RV2CU3
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64       STYRES3A           0018       008D7840 EXCLUSIVE    OWN

D U,VOL=RV2CU3
IEE457I 13.05.46 UNIT STATUS 002
UNIT TYPE STATUS        VOLSER     VOLSTATE
2557 3390 A-BSY-R       RV2CU3     PRIV/RSDNT
```

DFSMSdss full-volume dump serializes the VTOC of the source volume, but does not serialize the data sets on the volume. This ensures that the existing data sets are not deleted or extended, and new data sets are not allocated. However, there is an exposure in that the data in the existing data sets can be changed.

See Appendix B, "Data Integrity—Serialization" in the D*FSMS/MVS V1R5 DFSMSdss Storage Administration Reference,* SC26-4929, for further details about DFSMSdss serialization.

7.2 Sharing considerations (OS/390 2.8 and below)

An HFS can only be mounted read/write (R/W) on a single system, but it can be mounted read-only on multiple systems. Furthermore, on a single system, a file system can be mounted at only one directory at any time. This means that you cannot mount an HFS twice on the same system, whether in read/write mode or read-only mode.

As shown in Figure 74, if one system has an HFS mounted in read/write mode, no other system can mount the same HFS at the same time.

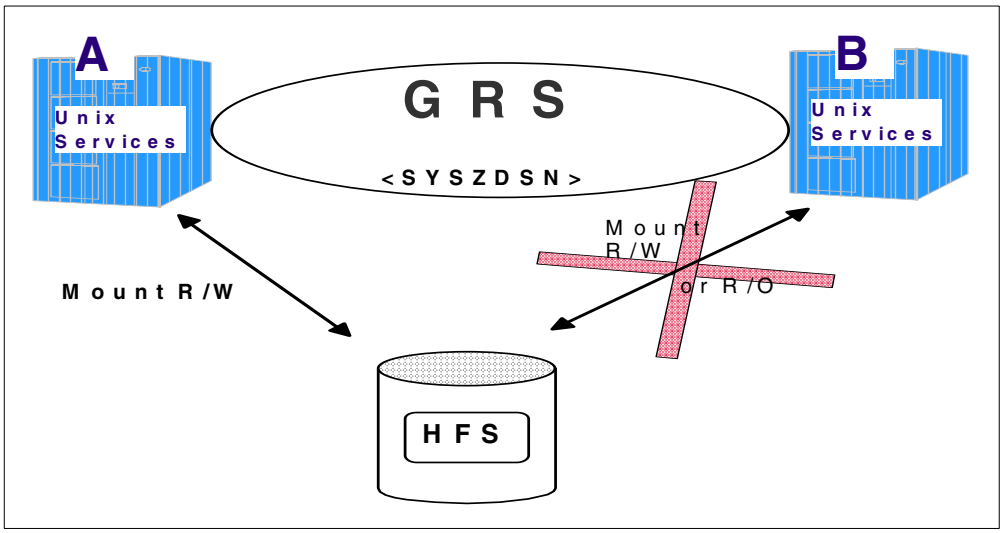


Figure 74. Cross system sharing restriction

Important information

An HFS mounted in read/write mode *cannot* be shared by multiple systems. If an HFS is mounted in read/write mode on one system, it may not be mounted either read/write or read-only on any other system.

SYSTEMS ENQ requests *must be* propagated for HFS data sets to ensure the integrity of your HFS data sets.

There were no changes in the rules for sharing an HFS across multiple systems in a Sysplex for DFSMS/MVS 1.5 when used with OS/390 2.8 or below.

You can only share an HFS data set across systems in a GRS complex if you mount an HFS data set in R/O mode on all systems.

However, the *OS/390 Version 2 Release 7 UNIX System Services Planning,* SC28-1890 contains the following advice:

> "We no longer recommend mounting an HFS in read-only mode and shared across all systems, because mounting the root file system in read mode will cause incomplete setup of certain OS/390 UNIX functions."

7.2.1 Mount integrity (write protection) enhancements

In the past, serialization was sometimes not handled properly across all systems in a multiple system environment. Most of these cases resulted in a broken index for the affected HFS, which means that the HFS was corrupted. Depending on which level or position of the index was broken, some or all files (data) in the HFS were no longer accessible.

Common symptoms for this situation were the message:

```
BPXF020I FILE SYSTEM MAY BE DAMAGED; RETURN CODE = 000000A2 ( x'A2':
EMVSPFSPERM - HFS encountered a system error)
```

This situation also had these errors:

- ABEND0F4 with reason code 145AA033 indicating index page damaged

  Or

- ABEND0F4 with reason code 05090035 when attempting to read beyond the highest record written.

There is no way to repair a broken HFS index. The only supported means to recover a damaged HFS is to restore from a backup copy.

Figure 75 shows such an environment. For example, systems B and C are in the same GRS complex. This means that ENQ SYSZDSN will be propagated to both systems. However, system A is not part of the complex; therefore, the appropriate ENQ will not be propagated to system A.

A sample scenario on DFSMS/MVS 1.4 (and older) systems might be as follows:

1. System C sends out a MOUNT R/W and enqueue on SYSZDSN and SYSDSN; System B sees the enqueue; the exclusive ENQ for SYSZDSN prevents system B from also mounting the HFS in R/W mode.
2. System A does not see the enqueue so it could also mount the same HFS in R/W mode.
3. System A issues a write.
4. System C also issues a write. This write will break the index, because now, both systems have different index structures in their own cache. The data (files) are probably inconsistent, too.

Note that it is not necessary to write, modify, or update an HFS to cause an index write. Just reading a file will cause the file's ATIME to be updated, resulting in an index write if the HFS is mounted R/W mode.

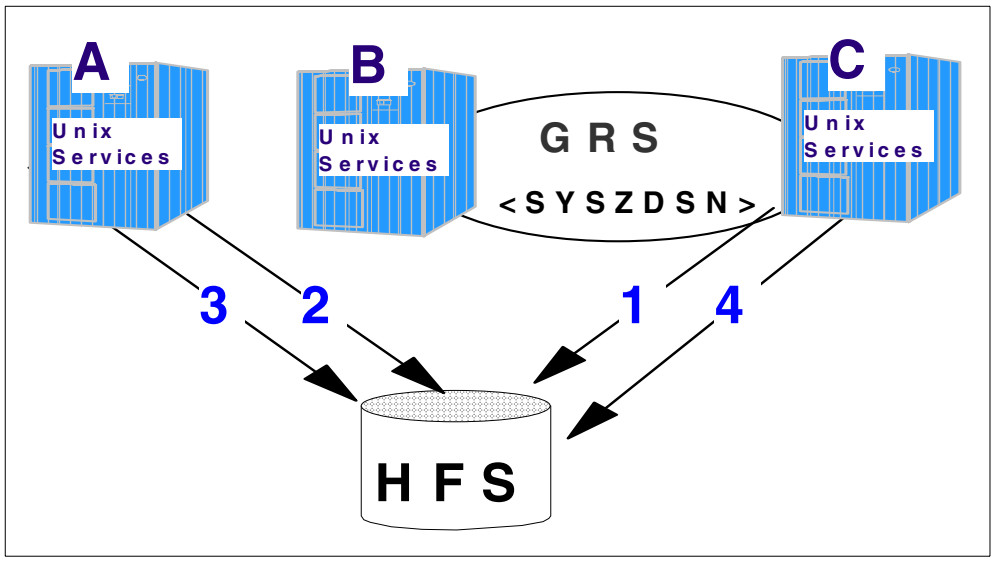


Figure 75. Sharing without write protection

In DFSMS/MVS 1.5, HFS services provide a new **write protection** mechanism to detect a second system that has also mounted the HFS in R/W mode.

Every system that mounts an HFS R/W writes the information into the HFS data set on DASD at mount time. Each system will also save the time stamp in its own storage. Figure 76 shows this in detail.

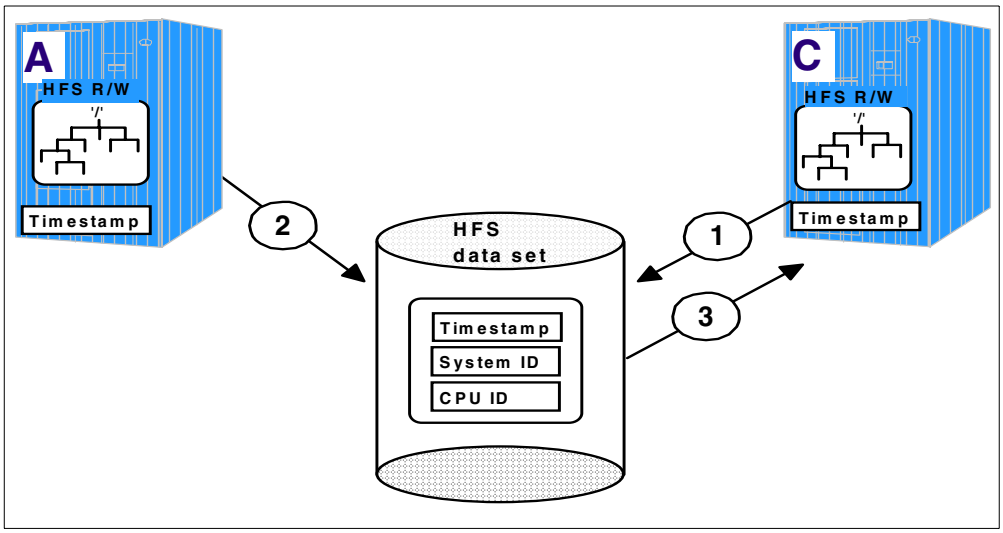


Figure 76. Write protection

1. System C writes its timestamp, system ID, and CPU ID at mount time into the HFS data set. System C will also save the timestamp in its storage.
2. Later on, System A writes its timestamp, system ID, and CPU ID at mount time into the HFS data set. It overwrites the information from system C.
3. At SYNC time, System C reads the information and compares it with that saved in its own storage.

   If the information does *not match*, the HFS will be flagged as bad on this system and all functions except unmount will not be executed and will return an error. The important point, in this case, is that no index nor data buffer will be written to DASD and the index structure of the HFS data set remains consistent. A console message will be issued to identify the other system:

   ```
   IGW020I HFS DATA SET dsname HAS BEEN MOUNTED R/W ON ANOTHER SYSTEM: sysname.
   FILE SYSTEM OPERATIONS ARE DISABLED UNTIL MOUNT.
   ```

   The associated message text explains the situation as well:

   System Programmer Response: Determine why an illegal R/W mount was done on another system and take steps to prevent future occurrences. Putting both systems into the same GRS ring is the recommended prevention mechanism. The file system can be reactived by unmounting it and then remounting it.

Future HFS calls will fail on System C, but they should not ABEND. An UNMOUNT and REMOUNT clears the status of the affected HFS. System A will be able to continue to access the HFS. Only system C is prevented from further access.

Note: The new write protection does *not* remove the restriction about sharing an HFS between two systems:

- An HFS mounted read/write *cannot* be shared by multiple systems before OS/390 2.9.
- SYSTEMS ENQ requests *must be* propagated to ensure the integrity of your HFS data sets.

You must still ensure that the HFS is serialized correctly by propagating the ENQ SYSZDSN to all other systems. The intention of the new write protection is to reduce the number of broken HFS data sets by detecting improper use of serialization of an HFS data set.

Figure 77 shows the benefits of the new write protection mechanism based on the sample and environment shown in Figure 75 on page 176.

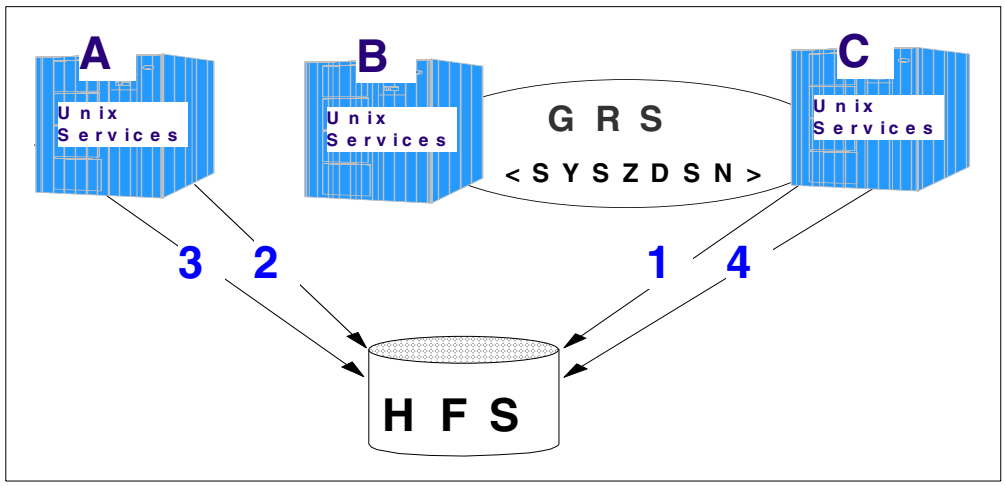


Figure 77. Sharing with write protection

What happens in a DFSMS/MVS 1.5 system with write protection is:

1. System C:
   - Mounts the HFS in R/W mode.
   - Writes its information into the HFS data set.
   - Enqueues on SYSZDSN and SYSDSN.

   Again, due to the GRS serialization, System B sees this enqueue.
2. System A does not see the enqueue, so it could:
   - Also mount the same HFS in R/W mode.
   - Overwrite the information on DASD with its own information
3. System A writes its buffer (index and data) at sync time to DASD.
4. At SYNC time, System C reads the information directly from the HFS data set on DASD and compares it with the information saved in its own storage.

   In this case, they do *not match*, so message `IGW020I` will be issued to identify the illegal mount from System A.

Note

The HFS will be flagged as bad, and *no* buffer (index nor files) will be written to DASD.

The *index* structure of the HFS *is still consistent*, because at sync time, we always write a consistent HFS to DASD.

However, the updated or changed files (data) could be inconsistent from system C's point of view.

After an UNMOUNT of the HFS on Systems A and C, you can MOUNT the HFS data set again on System C.

7.3 HFS sysplex sharing (OS/390 2.9 and above)

OS/390 2.9 UNIX System Services provides the capability for sysplex users to access data throughout the file hierarchy. Before 2.9, users logged onto one system in a sysplex had write access only to the file systems associated with their own system.

The shared HFS support provides these advantages:

- A common file system hierarchy on all systems. The root file system structure works in a sysplex and non-sysplex environment.
- Mount requests are sysplex-wide
- Writes to file systems from all systems in a participating group.
- *Rolling IPL's* to introduce new systems into the sysplex.
- Backout of systems from a sysplex.
- Greater availability of data in the event of a system outage by automatic recovery if a server system goes down.
- Multiple releases of the root file system (version root) are supported within a file system in a participating group.
- One common BPXPRMxx parmlib member for systems in a participating group.

A new file system structure was also introduced in OS/390 2.9 to support the shared HFS capability in a sysplex. See 9.3, "UNIX System Services (OS/390 2.9 and later)" on page 212 regarding the new file system structure.

7.3.1 Function shipping

OS/390 Unix System Services 2.9 implemented shared HFS support by using *function shipping*. Function shipping means that one system performs as a **server** to the other sharing systems (clients). A **client** system ships its requests to the related server(s) for each read/write shared HFS data set. Only the server system issues read and write requests for both types of data (files and metadata) to an HFS data set on a volume. The buffering and caching is also done within by the system which owns the file system.

By default, the system which mounts the HFS data set the first time automatically becomes the owning system. Figure 78 illustrates the server-client relationship.

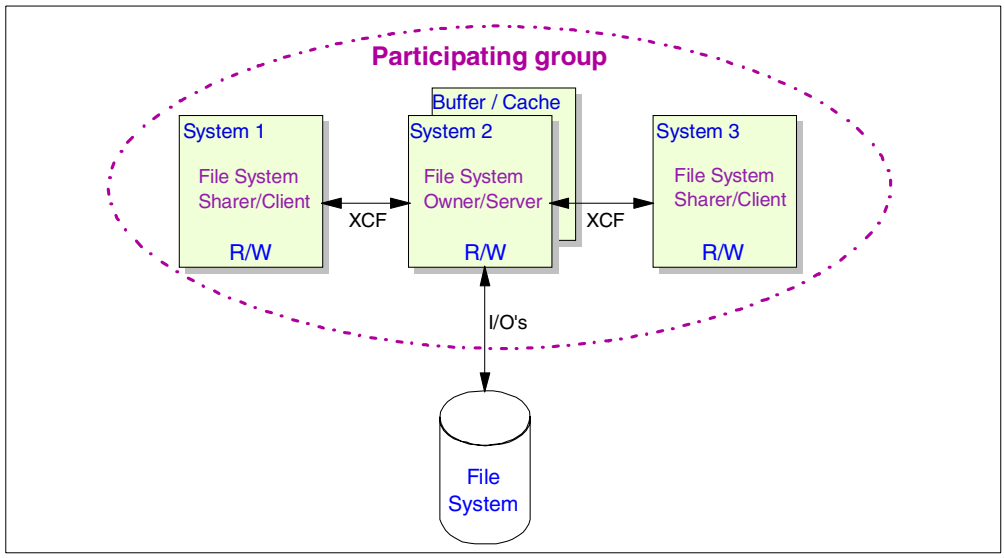


Figure 78. Shared HFS overview

The shared HFS support is implemented in the Virtual/Logical File System (VFS/LFS) layer within the OS/390 USS. Only one system can own a file system at a time within a participating group. However, each system can own different file systems at a time. As you can see in Figure 79, system 3 owns file system 2, but it can also update files on file system 1 owned by system 1 and vice versa. System 2 shares both file systems 1 and 2.

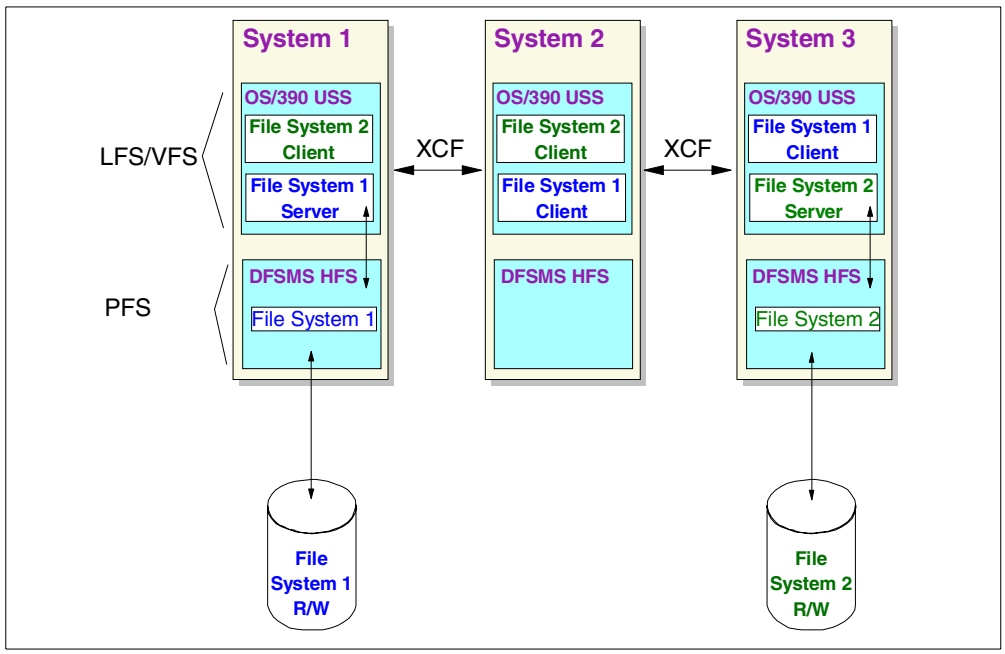


Figure 79. VFS and PFS relationship in read/write mode

Since HFS sharing is not implemented in the Physical File System (PFS) layer, represented by the DFSMS/MVS HFS functions, the same serialization rules apply to HFS data sets as existed before OS/390 2.9, if we leave the participating group.

Note: The PFS contains control blocks used to maintain and access the file system and its files and directories.

However, if you mount a file system in read-only mode, as shown in Figure 80 for *file system 1*, we have information about the HFS in the PFS on each system that has the HFS mounted. No function shipping will be done for file systems mounted read-only.

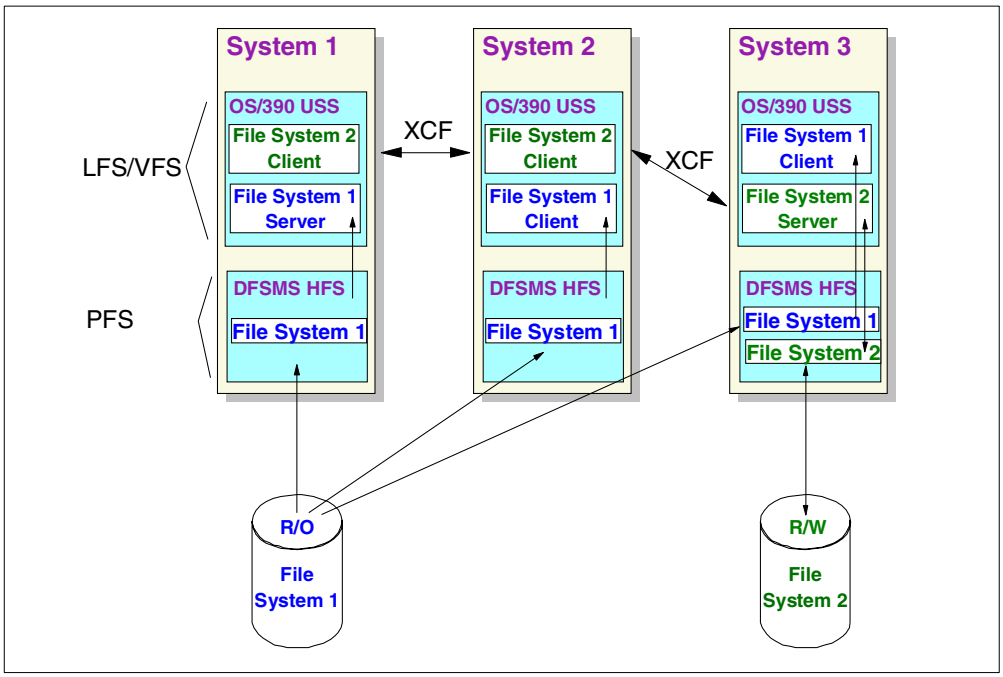


Figure 80. VFS and PFS relationship in read-only mode

Important Information

In order to share an HFS in read/write mode, all systems must be in the same participating (or SYSBPX) group in a sysplex.

In addition, all systems doing the read/write sharing must be at OS/390 2.9 or higher.

In order to share an HFS with pre-2.9 systems or with 2.9 systems outside the participating group, the HFS must be mounted read-only on *all* systems.

You will not be able to share an HFS between native MVS machines and second level VM guest machines.

If you are using shared HFS, REMOUNT is not supported. The way to remount a file system in a shared HFS environment is to UNMOUNT the file system and then MOUNT it again in the desired mode.

7.3.2 Sysplex sharing considerations

As you can see from Figure 68 on page 168 through Figure 70 on page 169, only one system has physical access to an file system although it is mounted read/write on other systems in a participating group.

Since the HFS sharing is implemented as function shipping, all systems in the participating group must exchange information around the participating systems in a sysplex (See also Figure 81).

The communication for the shared HFS support is implemented by using:

- XCF messaging services to inform all systems
  - When a system joins or leaves the SYSBPX group, and
  - To notify other systems that an update occurred in the system-wide mount table when a mount, unmount, chmount, quiesce or unquiesce request was issued.
- A coupling data set to track information about the participating systems and to have a sysplex-wide mount table. Note that a coupling *data set* is used, not a coupling facility structure.

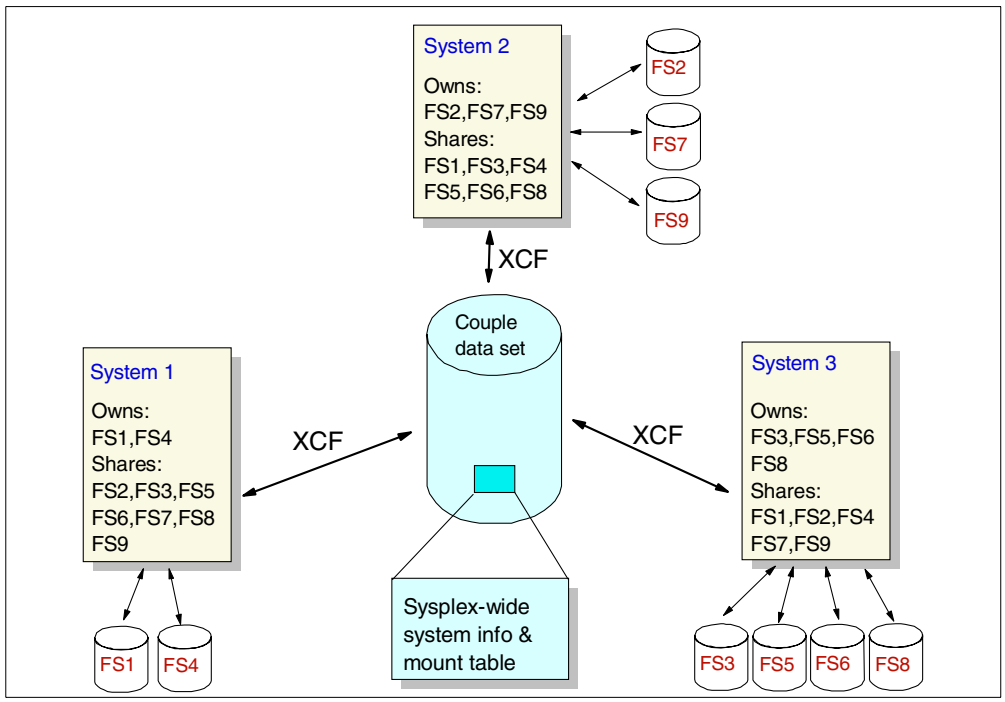


Figure 81. Shared HFS couple data set

Two new options are added to the mount requests to determine the owning system:

- **SYSNAME(sysname):** SYSNAME directs a mount request to the particular system on which a mount should be performed. This system will then become the owner of the mounted file system.
- **AUTOMOVE | NOAUTOMOVE:** AUTOMOVE indicates that ownership of the file system can automatically be moved to another system participating in shared HFS if the system that owns a file system goes down.

NOAUTOMOVE means that the file system will *not* be moved if the system goes down and therefore it is inaccessible from other systems.

See 3.3, “HFS PARMLIB and command enhancements” on page 56 for detailed information regarding the BPXPRMxx member and TSO MOUNT command enhancements.The AUTOMOUNT facility does not support the new options.

You may wish to force a mount of an HFS data set to a particular system for performance reasons. Assume that one system performs many updates to a particular file system which it doesn’t own and also assume that the owning system doesn’t perform updates to this file system. You can improve the performance by forcing the mount to the system performing the most updates. Thus, we avoid the overhead for function shipping.

Chapter 8, “Optimizing HFS performance” on page 185 provides further information regarding performance.

A **recovery** process for shared HFS data sets (mounted read/write) is also implemented. If an owning system fails, and AUTOMOVE was requested on the mount, takeover processing will be initiated to move the ownership of a file system to another system in the participating group. This system will then act as a server for this file system. This is also shown in Figure 82.

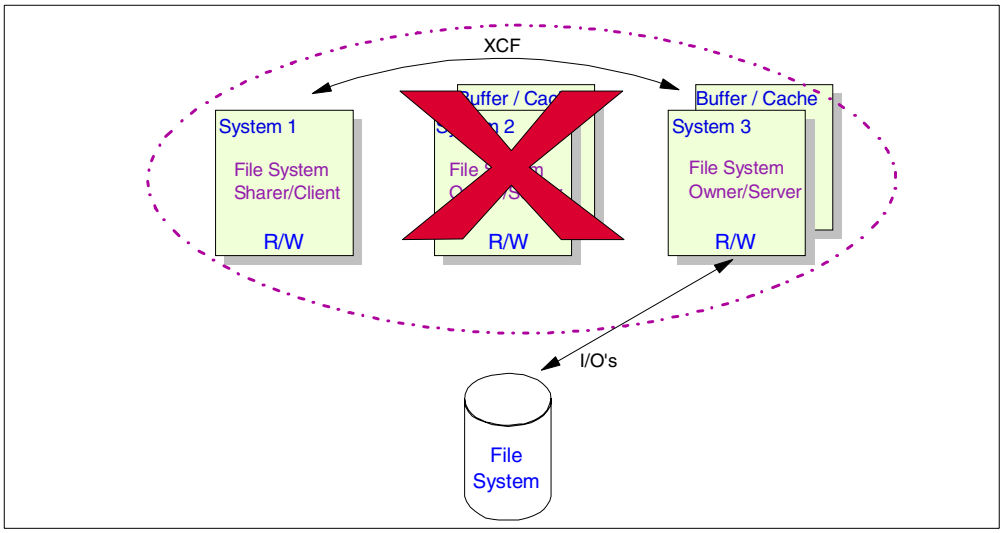


Figure 82. Recovering a file system.

The system attempts to perform recovery transparently to users. However, when it cannot do so, files will have to be closed and reopened.

You will find additional information regarding the file structure introduced with OS/390 2.9 and associated implementation information in:

- Chapter 9, “Implementing HFS for selected applications” on page 209
- Chapter 10, “HFS sysplex sharing implementation” on page 221
- *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890

Chapter 8. Optimizing HFS performance

DFSMS/MVS Version 1 Release 5 delivers a major restructuring of HFS. Throughput is improved dramatically because of reduced path lengths and improved buffering capabilities.

This chapter focuses on three sections:

- Section 8.1, “HFS performance restructure” on page 185 provides detailed information about the performance restructuring of HFS, that is, new deferred writes and new buffer management. Some performance data is shown in this section, too.
- In section 8.2, “RMF reporting enhancements” on page 192, the Resource Measurement Facility (RMF) provides new performance information about HFS. Additionally, this section explains how to set up RMF and the meaning of fields in the reports.
- Section 8.3 discusses how to optimize HFS performance.

Additional performance considerations are found in:

- Section 8.4, “Sysplex and HFS sharing” on page 204
- Section 8.5, “Distributed File System considerations” on page 206
- Section 8.6, “Using the copytree utility” on page 207

8.1 HFS performance restructure

The HFS performance restructure in DFSMS/MVS Version1 Release 5 implements a fundamental change in the way HFS data is written to disk, namely through the use of deferred writes and buffer management. These changes contributed to major performance improvements in elapsed time and CPU time.

This section provides detailed information about these enhancements to help you to understand the new HFS concepts.

8.1.1 HFS deferred writes

Prior to DFSMS/MVS 1.5, HFS hardened all metadata changes to disk, synchronous with user requests, and all file data changes in HFS cache were forced to be written to disk, synchronous with metadata hardening. For example, when an HFS file was created, the new file’s metadata as well as the parent directory’s metadata and data were written to disk before returning to the application.

This was true for all HFS functions that change HFS metadata, even though the POSIX and UNIX standards do not require metadata to be hardened upon return from functions that change metadata.

DFSMS/MVS 1.5 uses deferred writes, which are writes that are initially written to a system buffer rather than to disk. Deferred writes are then written to disk asynchronously by a *sync daemon*, without causing any immediate I/O delays to the application. Both data and metadata writes may be deferred. One of the most common write operations in UNIX is the updating the *last reference time stamp* (ATIME), which occurs whenever a file in a file system (HFS data set) is read. Since reads are typically more prevalent than writes, updating of the last

reference time stamp occurs very frequently, and elimination of such updates due to deferred writing can be very significant.

Instead of the immediate disk hardening of both data and metadata, a *sync daemon* will run at periodic intervals and write out both data and metadata changes that have occurred since the last time the sync daemon ran. Therefore, multiple metadata changes of an HFS file will be batched into a single I/O.

The sync daemon is part of the OMVS kernel address space, and it runs in the *Logical File System (LFS)* layer on UNIX System Services (USS). Figure 83 shows a diagram of the task and component structures related to HFS.

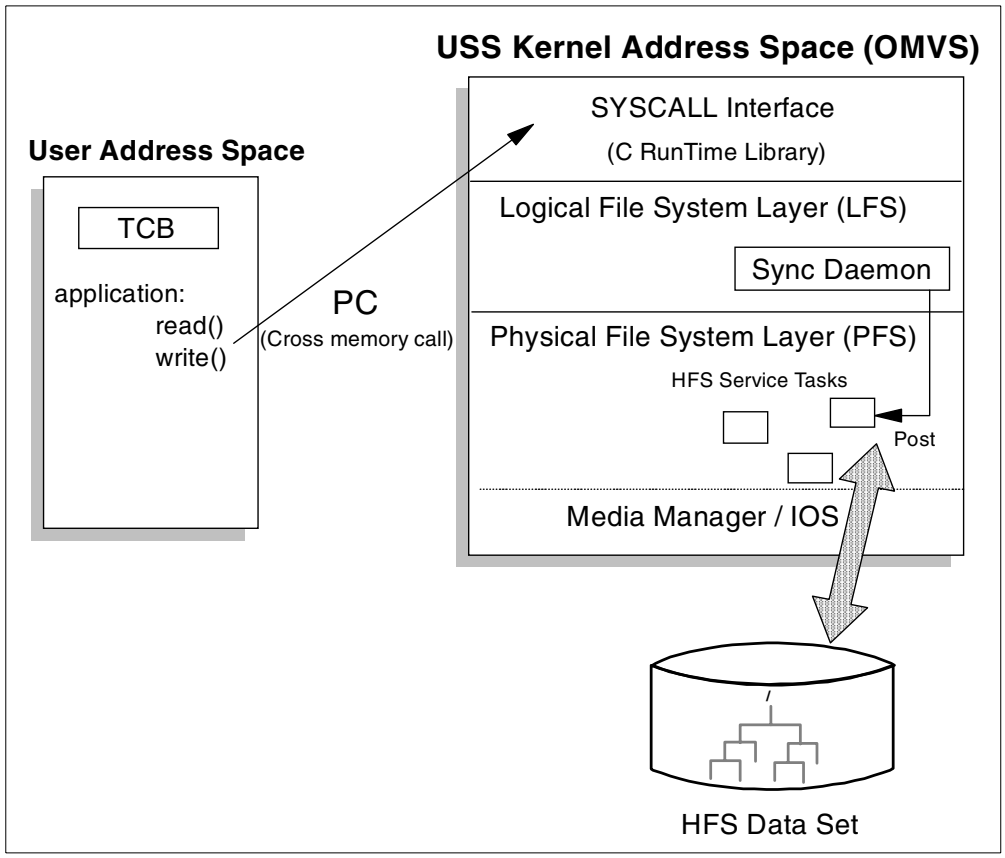


Figure 83. HFS task and component structure

As with other UNIX systems, OS/390 USS has a sync daemon which periodically writes all modified buffers to disk. The default sync interval is 60 seconds, but it is modifiable through the BPXPRMxx member in PARMLIB, or it may be specified differently for each file system when the file system is mounted.

The following examples show how to specify a sync interval in the BPXPRMxx PARMLIB member and also how to specify a sync parameter using /samples/mountx:

```
/samples/mountx -p 'sync(30)' /u/neils/test OMVS.HFS.NEILS.TEST
```

Or, in BPXPRMxx:

```
MOUNT FILESYSTYPE('OMVS.HFS.NEILS.TEST')
   MOUNTPOINT('/u/neils/test')
```

```
TYPE(HFS)
MODE(RDWR)
PARM('SYNCDEFAULT(30)')
```

Note: You can also mount an HFS through ISHELL and specify a sync interval.

Important Information

The sync process of the sync daemon will occur independently on each HFS file system. Even if sync intervals of all the HFS file systems are the same, the sync point of each HFS file system will be different on each mounted time.

When the system crashes before the metadata is hardened, newly created files will be missing and recently removed files will still be present. The integrity of the file system is preserved because of the index shadowing.

Figure 84 below shows an example when *System Crash 1* occurs between Sync *Point N and N+1*. The results from *Activity 1 and 2* are not hardened to the file system. The conditions of the file system are set back to the conditions on *Sync Point N.*

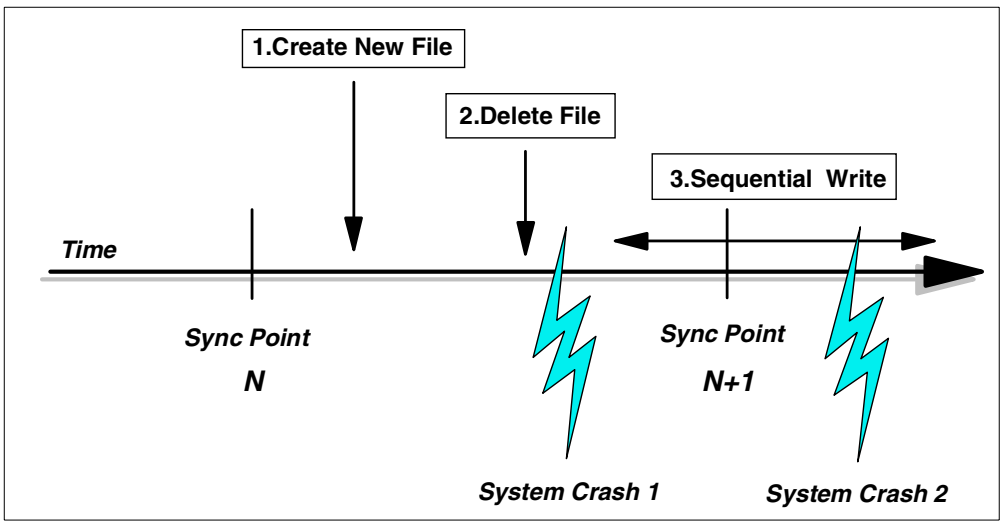


Figure 84. Example of sync daemon and system crash

In case of a long process across the sync point such as a *Sequential Write of Activity 3*, if *System Crash 2* occurs between *Sync Point N+1 and the end of process,* the condition of the file is set back to the state it had at N+1.

8.1.2 HFS caching and buffering

In DFSMS/MVS 1.4, the HFS component calls another component to obtain index (such as metadata) buffers and perform lookaside on the file data.

With DFSMS 1.5, caching and buffering for file and index data are dramatically improved by restructuring and consolidating the components. This consolidation leads to reduced pathlength which results in a performance gain.

Figure 85 on page 188 shows the address spaces and data spaces of DFSMS/MVS 1.5, (consisting of HFS caching and buffering functions) compared to DFSMS/MVS 1.4.

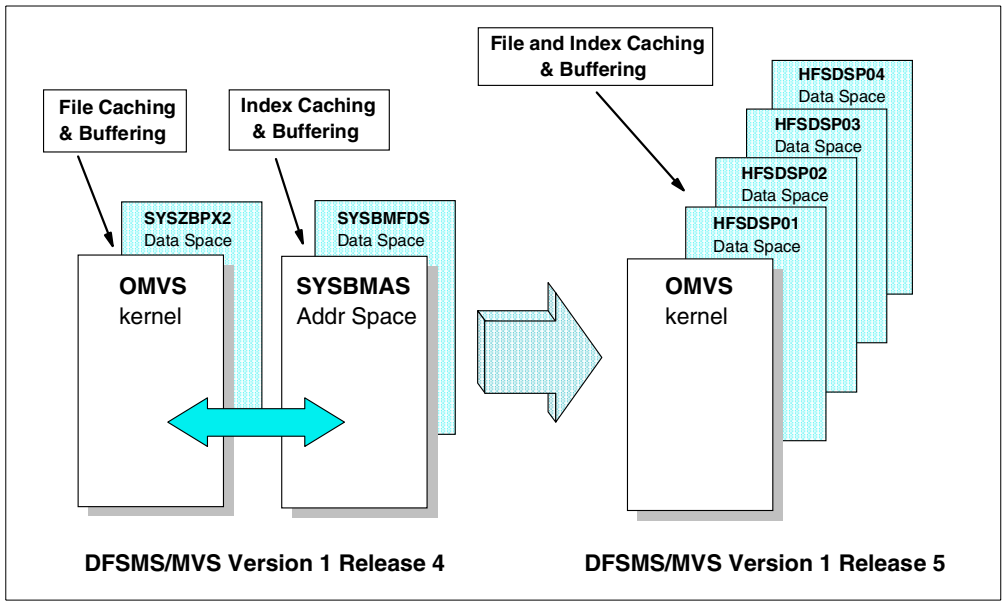


Figure 85. Restructuring of file and index caching and buffering

In DFSMS/MVS 1.4, file data is cached and buffered in one data space owned by the OMVS address space. Index data (such as metadata) and some small file data is cached and buffered in one data space owned by SYSBMAS which is also used by PDSE. Furthermore, large file data is read into the OMVS address space itself. If 2GB of these data spaces is exhausted, the HFS component gives an 0F4 abend and fails the request. The user has no control over the cache and buffer sizes.

In DFSMS/MVS 1.5, a user can control these pool sizes using the confighfs shell command. Two confighfs settings, VIRTUAL and FIXED are provided to control the virtual storage and real storage allocated to HFS to be used for caching and buffering. The default value for VIRTUAL is set to 50% of real storage and the default value for FIXED is set to zero. For more information about the confighfs shell command, please refer to 3.3.5, “confighfs shell command” on page 68.

The HFS component initially allocates four 2GB data spaces for four different buffer pools. These data spaces are owned by the OMVS address space and are called HFSDSP01 through HFSDSP04. File and index data (such as metadata) are initially cached and buffered in the HFSDSP01 data space.

The HFS buffers are divided into four pools: the 4KB pool, the 16KB pool, the 64KB pool and the 256KB pool. If one of these data spaces is exhausted, another data space may be automatically allocated by HFS.

In the case of a sequential I/O, the data spaces are divided into the following pools:

- 4KB pool: For small size files (equal or less than 4KB)

- 16KB and 64KB pool: For intermediate size files (greater than 4KB but equal or less than 64KB)
- 256KB pool: For large size files (greater than 64KB)

The buffer selection algorithm is summarized as follows:

- **Preference scheme of HFS buffer page selection**

  When an application needs to acquire an HFS buffer, HFS chooses a buffer based on a preference scheme. Since the purpose of allocating a new buffer is generally to do I/O, which requires that the buffer be page-fixed, the first preference is to use a permanently fixed buffer. (This statement is obvious for reads, but even deferred write will cause an I/O in the near future.) If no permanently fixed buffers are available, the second preference is to use a pageable buffer that is already backed by real storage if one is available. If all such buffers are in-use, then a new buffer is acquired, and the first time that the buffer is touched, the Real Storage Manager (RSM) will allocate some real storage.

- **Number and size of HFS buffer selection**

  The 4KB pool is used for all metadata and for all files that are less than 4KB in size. For sequential reads, HFS chooses the minimum pool size that it will allow the entire file to be read into a single buffer. For random reads and writes, HFS chooses the pool size based on the number of bytes being requested and the offset from a page boundary, choosing the smallest buffer such that I/O can be satisfied using a single buffer. For sequential writes, the algorithm is more complicated. HFS uses gradually increasing buffer sizes up to the largest 256KB buffer size. The number of buffers that HFS will use to cache a file is generally limited to 4, which provides for the caching of up to 1MB. More buffers may be allocated per file if the VIRTUAL parameter exceeds 2GB, but it is generally recommended that VIRTUAL not be set larger than 2GB.

8.1.3 Performance data

This topic shows the measurement results for single-user file system reads and writes. These measurements clearly demonstrate the dramatic reductions in CPU time, as well as the reductions in elapsed time.

Note: These results were obtained in a controlled environment. Actual customer results may vary, depending on their environments. Please also note that the vertical axes are logarithmic, not linear.

8.1.3.1 Small file performance

In both sequential read and write cases of small files, CPU and elapsed time are dramatically reduced. As shown in Figure 86 on page 190, CPU time in DFSMS Version 1 Release 5 equals its elapsed time, which means there is no I/O, and all data is cached in the buffer pool. Previous DFSMS/MVS releases update the last reference time stamp synchronously, even for read accesses.

Note: The CPU and I/O time of the sync daemon process (asynchronous I/O of a file) is charged to the OMVS kernel address space.

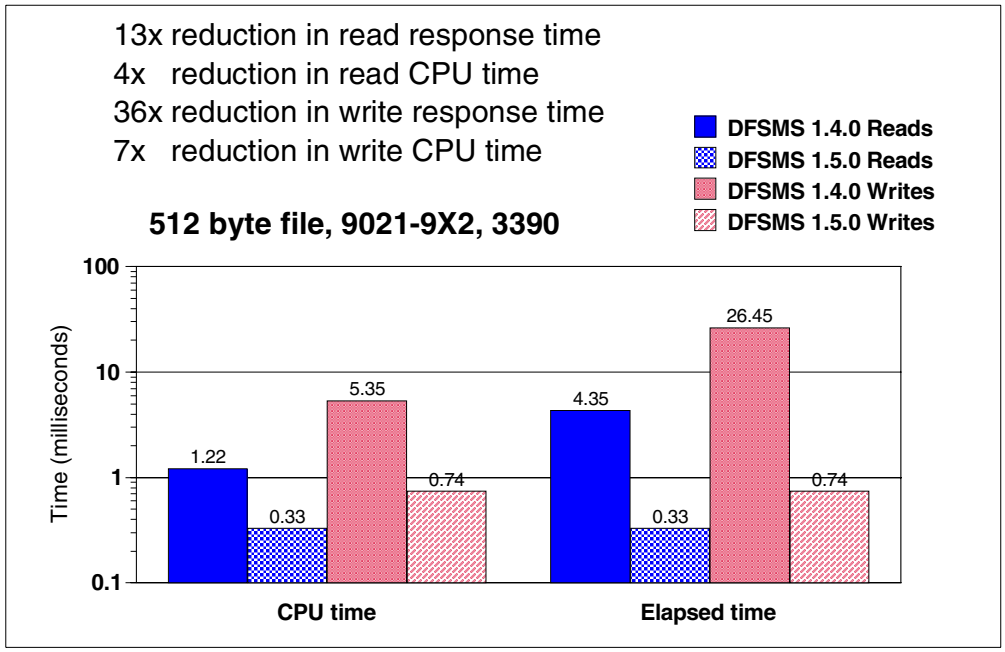


Figure 86. Small file performance improvements

8.1.3.2 Single user read/write performance

The graphs in Figure 87 and Figure 88 on page 191 show the CPU and elapsed time of various file sizes for both sequential read and write. The HFS improvements in DFSMS/MVS 1.5 reduce the CPU time dramatically for all file sizes.

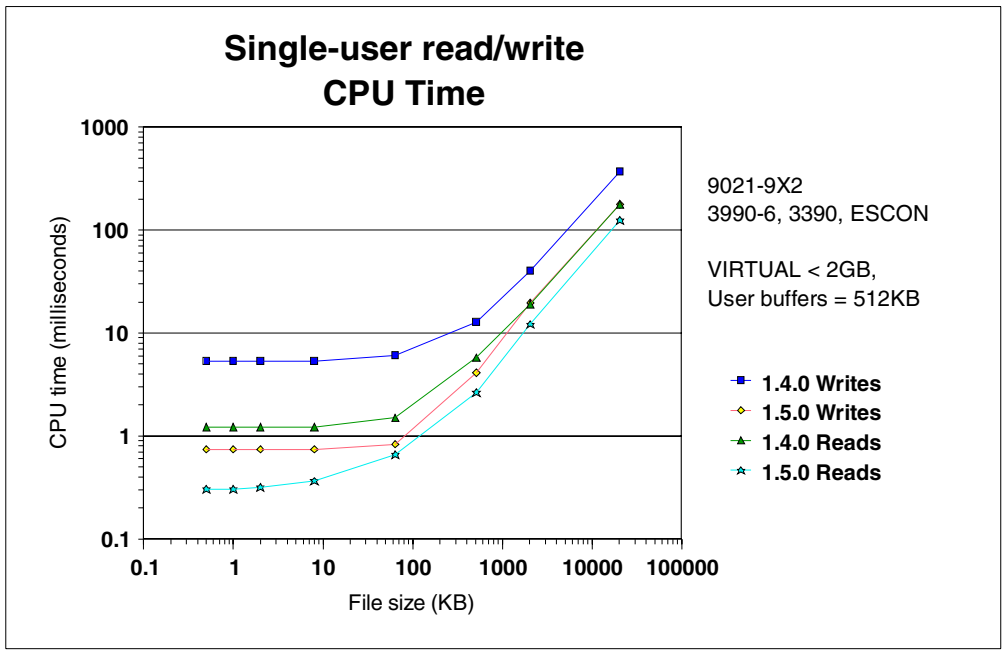


Figure 87. Single user read/write CPU time

In the elapsed time measurements shown in the previous graph, the reason that files larger than 1MB performed worse than the files smaller than 1MB is that synchronous I/O to write data pages occurs at that time, since the maximum buffer size of a file is 1MB (256KBx4).

You will find that the CPU time corresponds to its elapsed time in the best performance case.

Important Information

The typical situations when synchronous I/Os occur on HFS are as follows:

- Synchronous writes of *data pages* are done whenever all the HFS buffers of the file are filled.
- Synchronous reads of *data pages* are done whenever there is a cache miss.
- Synchronous writes of *metadata and data pages* are done whenever an fsync function is issued. For information about the fsync function, refer to 8.3.3, “fsync call considerations” on page 204.

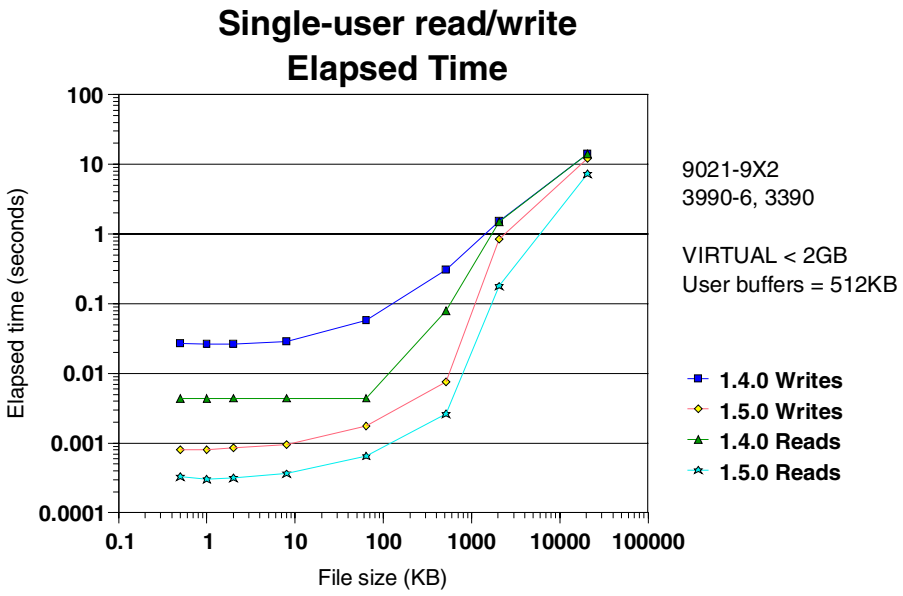


Figure 88. Single user read/write elapsed time

Note

The DFSMS/MVS 1.5 announcement letter states that the performance border for the new HFS support tends to be 512KB. This is not correct. CPU and elapsed time reductions are seen at all sizes. However, the most significant elapsed time reductions are for files that are 1MB or smaller.

8.2 RMF reporting enhancements

With OS/390 2.7, new RMF (Resource Measurement Facility) reports are available for Monitor II and the Postprocessor. These reports provide performance information on HFS which allows you to tune your system and use HFS resources better. You can use the storage information and utilization statistics in the new reports to determine the optimum size of the HFS buffer storage.

This section provides information about setting up RMF and the meaning of fields in the report.

8.2.1 New RMF reports for HFS

Two new reports offer performance information on HFS:

- The Monitor II Hierarchical File System Statistics (HFS) report shows data for performance analysis related to I/O activities in the OS/390 UNIX environment.
- The Postprocessor Hierarchical File System Statistics (HFS) report shows data for performance analysis and capacity planning.

8.2.2 Setting up RMF Monitor III

Data gathering for HFS file statistics in the Postprocessor report will be performed in the Monitor III gatherer session. The Monitor III gathers SMF record 74 subtype 6. A new option, *HFSNAME,* is available for Monitor III data gathering to define the appropriate files. If you do not set option HFSNAME, you only get the HFS global statistics report of the Postprocessor.

For example, in Figure 89, to gather the appropriate files, HFS.SC64.STYRES1 and HFS.SC64.STYRES2, the following statement should be added in ERBRMFxx of Monitor III.

```
HFSNAME(ADD(OMVS.SC64.STYRES1),
        ADD(OMVS.SC64.STYRES2))
```

Figure 89. Example of ERBRMFxx for Monitor III

You can dynamically enable or disable this option by using these OS/390 operator commands:

```
F RMF,F III,HFSNAME(ADD(OMVS.SC64.STYRES1))

F RMF,F III,HFSNAME(DEL(OMVS.SC64.STYRES1))
```

8.2.3 HFS Postprocessor report

After data gathering for HFS file statistics is performed in the Monitor III gatherer session, the Postprocessor provides HFS global statistics data and HFS file statistics data.

For example, in Figure 90, to produce the report you should specify the following option in the RMF REPORTS command.

```
REPORTS(HFS,...)
```

Figure 90. Example of REPORTS command

The HFS report consists of two parts:

- **HFS Global Statistics Report**

  The first part of the HFS report provides overall data about the I/O activity to HFS files and gives statistics about the various buffer pools that have been defined. The report can be used as an entry point for performance investigation and capacity planning.

- **HFS File System Statistics Report**

  The second part of the report is based on data gathered for specific file systems. You can get the statistics about I/O activities and the internal structure (index) of the HFS files.

Both parts of the report can help you to make best use of your resources.

8.2.3.1 HFS global statistics

An example of the HFS global statistics report is shown in Figure 91.

```
                                H F S   G L O B A L   S T A T I S T I C S

          OS/390                    SYSTEM ID TSC3             DATE 07/05/1999            INTERVAL 01.00.000
          REL. 02.07.00             RPT VERSION 2.7.0          TIME 20.52.00              CYCLE 5.000 SECONDS
       --- STORAGE LIMITS (MB) --                       ----- FILE I/O ----  --- METADATA I/O --
                                                             COUNT     RATE       COUNT     RATE
       VIRTUAL      MAX       128
                    USE     7.007      CACHE                138    2.300          12    0.200
       FIXED        MIN        32      DASD                   6    0.100           2    0.033
                    USE     0.156      HIT RATIO          95.83                85.71
-------------------------------------- BUFFER POOL STATISTICS ----------------------------------------
  POOL       NUMBER    BUFFER   -------- POOL SIZE --------      DATA      ---------- I/O ACTIVITY ---------
NUMBER      BUFFERS      SIZE     PAGES     BYTES    %FIXED    SPACES        TOTAL        FIXED       %FIXED
     1          202         1       202      808K         9         1           11            8           73
     2            6         4        24       96K        16         1            0            0            0
     3            6        16        96      384K        16         1            0            0            0
     4           23        64      1472     5888K         0         1          122          114           93
```

Figure 91. Example of HFS global statistics

The HFS global statistics report shows you a summary of HFS activity for the entire system and each buffer pool. It can then be used for determining the VIRTUAL and FIXED values. The important point to note is that the CACHE, DASD, and HIT RATIO fields in the FILE I/O section show only the statistics for sequential I/O. Statistics for random I/O are not currently provided.

The field descriptions shown in the report are listed in Table 8.

Table 8. RMF field descriptions - HFS global statistics

Field Heading	Meaning
Storage Limits - All fields are given in megabytes and show the values at interval end.	
VIRTUAL MAX	Value of VIRTUAL(MAX) parameter.
VIRTUAL USE	Total amount of virtual storage assigned to I/O buffers.
FIXED MIN	Value of FIXED(MIN) parameter.
FIXED USE	Total amount of permanently fixed storage assigned to I/O buffers. This number is included in the VIRTUAL USE field.
File I/O - The fields are given as COUNT and RATE (count per second).	
CACHE	The first page of a data file was requested and found in virtual storage (cache).
DASD	The first page of a data file was requested and not found in virtual storage, and an I/O was necessary.
HIT RATIO	Percentage of cache-found requests based on total number of requests.
Metadata I/O - The fields are given as COUNT and RATE (count per second).	
CACHE	The metadata for a file was found in virtual storage during file lookup.
DASD	The metadata for a file was not found in virtual storage during file lookup, and an index call was necessary which may have resulted in an I/O.
HIT RATIO	Percentage of cache-found requests based on total number of requests.
Buffer Pool Statistics	
POOL NUMBER	HFS defines four buffer pools for processing. This number is used to refer to each of these pools.
NUMBER BUFFERS	Number of buffers in this buffer pool currently residing in virtual storage.
BUFFER SIZE	Size of each buffer in this pool (in pages).
POOL SIZE - PAGE	Size of this buffer pool currently in virtual storage (in pages).
POOL SIZE - BYTES	Size of this buffer pool currently in virtual storage (in bytes).
POOL SIZE - %FIXED	Percentage of the size of the buffers which is permanently fixed.
DATA SPACES	Number of data spaces comprising this buffer pool.
I/O ACTIVITY - TOTAL	Total number of buffers in this buffer pool for which I/Os were issued. This is not necessarily the number of actual I/Os issued since multiple buffers can be written in a single I/O request.

Field Heading	Meaning
I/O ACTIVITY - FIXED	Number of times a buffer was already fixed prior to an I/O request in this buffer pool.
I/O ACTIVITY - %FIXED	Percentage of fixed I/Os.

8.2.3.2 HFS file statistics

The example of an HFS file statistics report is shown in Figure 92.

```
                          H F S   F I L E   S Y S T E M   S T A T I S T I C S

          OS/390                    SYSTEM ID TSC3             DATE 07/05/1999           INTERVAL 01.00.000
          REL. 02.07.00             RPT VERSION 2.7.0          TIME 20.51.00             CYCLE 5.000 SECONDS
--- ALLOCATION (MB) --                    ----- FILE I/O ----  --- METADATA I/O --  ---- INDEX I/O ----     ---- INDEX EVENTS ---
                SIZE                       COUNT     RATE       COUNT     RATE       COUNT     RATE                      COUNT
FILE SYSTEM NAME: OMVS.OS270.TSC3.ROOT
MOUNT DATE: 07/05/1999   TIME: 20:03:16
SYSTEM             668     CACHE                0    0.000          0    0.000          8    0.133     NEW LEVEL           0
DATA               591     DASD                 0    0.000          0    0.000          0    0.000     SPLITS              0
ATTR. DIR        2.808     HIT RATIO         0.00               0.00              100.00               JOINS               0
                           SEQUENTIAL           0    0.000
CACHED           0.089     RANDOM               0    0.000
FILE SYSTEM NAME: OMVS.TSC3.S067157
MOUNT DATE: 07/05/1999   TIME: 20:47:50
SYSTEM              92     CACHE               28    0.467          0    0.000         22    0.367     NEW LEVEL           0
DATA                53     DASD                 3    0.050          4    0.067          1    0.017     SPLITS              0
ATTR. DIR        0.324     HIT RATIO        90.32               0.00               95.65               JOINS               0
                           SEQUENTIAL           3    0.050
CACHED           0.000     RANDOM               0    0.000
```

Figure 92. Example of HFS file system statistics

The HFS file statistics report shows the statistics for each HFS file system. The important points to note are that the CACHE, DASD, and HIT RATIO fields in the FILE I/O section show only the statistics for *sequential* I/O. If your applications use random I/O to certain HFS file systems those caching statistics are not currently available.

The field descriptions shown in the report are listed in Table 9.

Important Information

HFS File System Statistics shows Metadata I/O and Index I/O separately.

Metadata I/O means I/O on behalf of an RNODE, which represents one file in an HFS. An RNODE is created during the first access of a file and contains, among other things, a pointer to the attribute directory. RNODEs reside in the OMVS kernel address space.

Index I/O means I/O activities to all the index data and metadata on disk or in a data space, and metadata on disk or cached in a data space.

Table 9. RMF field descriptions - HFS file system statistics

Field Heading	Meaning
FILE SYSTEM NAME	The name of the HFS file system which has been selected for reporting.
MOUNT DATE and TIME	Date and time when the selected file system was mounted.
Allocation - All fields are given in megabytes.	
SYSTEM	Amount of storage allocated to this file system.
DATA	Amount of storage internally used within HFS for data files, directories and HFS internal structures like the attribute directory (AD).
ATTR. DIR	Amount of storage used for the attribute directory (AD). This number is included in the DATA field. The attribute directory is the internal HFS structure (index) which contains attribute information about individual file system objects as well as attributes of the file system itself.
CACHED	Amount of data buffer storage cached by this file system.
File I/O - The fields are given as COUNT and RATE (count per second).	
CACHE	The first page of a data file was requested and found in virtual storage (cache).
DASD	The first page of a data file was requested but was not found in virtual storage (cache) and an I/O was necessary.
HIT RATIO	Percentage of cache-found requests based on total number of requests.
SEQUENTIAL	Sequential file data I/O requests. A sequential I/O is one of a series of I/Os to read or write a data file, where the first I/O started at the first byte of the file and each subsequent I/O was for the next sequential set of bytes.
RANDOM	Random file data I/O requests. A random I/O is an I/O that does not read or write the start of a file, and was not preceded by an I/O that read or wrote the immediately preceding set of bytes.
Metadata I/O - The fields are given as COUNT and RATE (count per second).	
CACHE	The metadata for a file was found in virtual storage (cache) during file lookup.
DASD	The metadata for a file was not found in virtual storage during file lookup and an index call was necessary which may result in an I/O.
HIT RATIO	Percentage of cache-found requests based on total number of requests.

Field Heading	Meaning
Index I/O - The fields are given as COUNT and RATE (count per second).	
CACHE	Index page read/write hits.
DASD	Index page read/write misses.
HIT RATIO	Percentage of cache-found requests based on total number of requests.
Index Events	
NEW LEVEL	Number of times HFS added a new level to its index structure. The index statistics are relative to all of the indices in the HFS data set. The attribute directory (AD) is one index.
SPLITS	Number of times an index page was split into two pages because new records were inserted. This gives an idea of how much insertion activity there has been for the index structure.
JOINS	Number of times HFS was able to combine two index pages into one, because enough index records had been deleted in the two pages.

8.2.4 Setting up RMF Monitor II

After you start RMF Monitor II, you can enter the Monitor II Primary Menu from your ISPF menu to see the HFS statistics report.

When you select Monitor II on the RMF Primary Menu, you get the Monitor II Primary Menu. As you can see in Figure 93, you can go from here to selection 2*, I/O Subsystem*.

```
                    RMF Monitor II Primary Menu                OS/390 2.7.0 RMF

Enter selection number or command on selection line.



  1 Address Spaces      Address space reports
  2 I/O Subsystem       I/O Queuing, Device, Channel, and HFS reports
  3 Resource            Enqueue, Storage, SRM, and other resource reports

  L Library Lists       Program library information
  U User                User-written reports (add your own...)


                              T TUTORIAL    X EXIT



          5647-A01 (C) Copyright IBM Corp. 1994, 1999. All Rights Reserved
                     Licensed Materials - Property of IBM


Selection ===>2
```

Figure 93. RMF Monitor II Primary Menu

From this panel in Figure 94, you can access information about HFS from option 5, *HFS Statistics*.

```
                    RMF Monitor II I/O Report Selection Menu

Enter selection number or command on selection line.

  1 CHANNEL          Channel path activity
  2 IOQUEUE          I/O queuing activity

  3 DEV              Device activity
  4 DEVV             Device activity by volume or number

  5 HFS              Hierarchical file system statistics


Selection ===> 5
```

Figure 94. Monitor II I/O Report Selection Menu

When you select option 5 for the first time, the panel *HFS Report Options* is shown as in Figure 95. You can select one of the listed HFS data sets to be monitored and then press PF3.

```
RMF Monitor II - HFS Report Options         Line 1 of 9

Select (S) or fill-in a file system name. To exit press END.

Selected file system name:
Number of mounted file systems: 11          Display: YES    (YES/NO)

You can use FIND to search for a specific HFS file system name.

Sel  HFS File System Name
 _   OMVS.HFS.TEST1
 _   OMVS.OS270.ETC
 _   OMVS.OS270.ROOT
 _   OMVS.SC64.STYRES1
 S   OMVS.SC64.STYRES2
 _   OMVS.SC64.STYRES3
 _   OMVS.SC64.STYRES4


Command ===>                                                Scroll ===> PAGE
```

Figure 95. Monitor II HFS Report Options

The *HFS file system statistics* for OMVS.SC64.STYRES2 are shown in Figure 96.

The next time you display these panels, the *HFS Report Options* screen is not shown. If you want to change this option later, you can enter the *RO* command on the *HFS file system statistics* screen.

```
                        RMF - HFS File System Statistics            Line 1 of 12

                        CPU= 23     UIC=254 PFR=   0         System= SC64 Total

File System Name: OMVS.SC64.STYRES2
Mount Date: 07/01/1999   Time: 16:21:39                 Elapsed Time: 05:51:12

----------- Allocation (MB) ----------   ------------ Index Events ------------
System          70    Data          24   New Level        0
Attr. Dir    0.019    Cached     0.000   Splits           0    Joins          0

              ----- File I/O ----    --- Metadata I/O --    ---- Index I/O ----
                   Count     Rate         Count     Rate         Count     Rate

Cache                  0    0.000             1    0.000          1116    0.053
DASD                   5    0.000             1    0.000             1    0.000
Hit Ratio           0.00                   50.00                 99.91
Sequential             0    0.000
Random                 0    0.000



Command ===>                                                 Scroll ===> PAGE
```

Figure 96. Monitor II HFS File System Statistics

The meanings of the fields displayed for the *HFS file system statistics report* are the same as those displayed in the Postprocessor report. Refer to 8.2.3.2, “HFS file statistics” on page 195 for more details.

8.3 Optimizing HFS performance

This section provides hints and tips that may help you to optimize HFS performance.

There are five options available to tune HFS I/O performance:

1. Setting SYNCDEFAULT or SYNC parameter
2. Setting VIRTUAL and FIXED parameters
3. Rearranging HFS data sets
4. Dividing one HFS data set into several smaller HFS data sets
5. Setting each product’s parameters related to their own caching

We will only discuss the first two considerations in detail as these are specific to HFS. The other recommendations are general I/O tuning recommendations.

8.3.1 SYNCDEFAULT and SYNC setting

In general, setting a long sync interval on UNIX systems would cause a performance impact to the entire system, whereas the impact on S/390 would not be as severe. There are several reasons for this, namely, the disk devices and I/O subsystems on S/390 are more sophisticated, providing fast interfaces, multiple paths, and a large amount of cache in a control unit. Also, OS/390 provides a System Resource Manager (SRM) which optimizes resource usage for multiple tasks running concurrently. It is for these reasons that setting a long sync interval is not as detrimental to system performance on S/390 as on other UNIX systems.

If you set a sync interval to a non-zero value, you will get much better performance than if you set a sync interval of zero. A different sync interval may be specified for each file system. A sync interval of zero signifies that deferred writes are not used. However, since an application may still use deferred writes and control the writing of data by using the fsync call, it is generally recommended that a non-zero sync interval be used. For more information about the fsync call, refer to 8.3.3, “fsync call considerations” on page 204.

Figure 97 shows the effect obtained by setting various sync interval lengths.

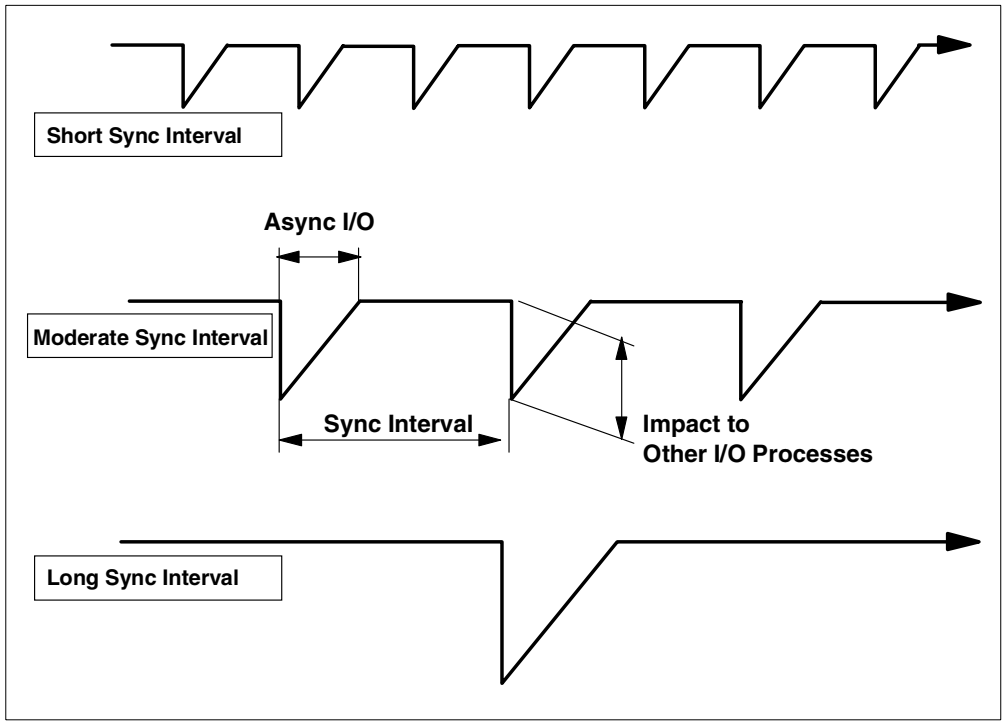


Figure 97. Effect of various sync interval lengths

Next, we will discuss the advantages and disadvantages of setting the sync interval too high or too low.

8.3.1.1 Setting a short sync interval

One reason for lowering the sync interval is that some control units do not perform well if they are flooded with many writes all at once. This may happen if the channel speed is higher than the rate that the control unit can destage data out of its cache or non-volatile storage and if the size of the non-volatile storage is small.

A second reason for lowering the sync interval is to avoid high variations in response times. While the deferred writes are executing, read operations cannot get to the devices. By lowering the sync interval, the interference that writes cause can be spread more evenly over time.

Finally, a shorter sync interval minimizes data loss in the event of a system failure.

8.3.1.2 Setting a long sync interval

Before thinking about a performance influence, you should consider that setting a long interval will increase the possibility of losing updated data if the system crashes.

From a performance standpoint, the benefit of setting longer intervals will depend on how frequently the same file data or metadata is updated. It is more likely that the same metadata pages are repeatedly updated than the file data pages, but it would depend on the application. If pages are repeatedly updated, then a longer sync interval would reduce the number of data and I/O operations.

On the other hand, a long sync interval would decrease the number of available metadata pages to the HFS file system during the interval, because the *index shadowing* holds updated metadata pages until the next sync point. For nearly-full HFS data sets, this is an important consideration. You can analyze the usage from the following sources:

- ATTR.DIR field of HFS File System Statistics report from RMF
- Attribute pages field of confighfs -q command output
- df -P USS command output

A long interval increases the possibility that a larger percentage of buffers will be filled with modified data. This may lead to a lower read hit ratio in some cases.

When you mount HFS file systems in BPXPRMxx in SYS1.PARMLIB, the mount times of the HFS file systems will be nearly the same. If HFS file systems with the same long sync interval are allocated on the same device, all I/O operations to each of the file systems will be overlapped. To avoid this I/O contention, you can do the following:

- Disperse the HFS data sets to other volumes
- Move HFS data sets to faster devices
- Stagger mount commands by using a /etc/rc shell script with mountx and sleep commands instead of the BPXPRMxx

8.3.2 VIRTUAL and FIXED setting

In DFSMS 1.5, the way that HFS manages I/O buffers and caches data in the processor is very different from that in DFSMS 1.4. For the most part, these changes are transparent to the system programmer. However, there are two new controls which the system programmer can use to tune the system in terms of managing I/O buffers and caching data. There are also some new performance reports that provide better awareness of what needs to be tuned.

The two new controls are the *VIRTUAL* and *FIXED* parameters, which may either be specified in the BPXPRMXX in PARMLIB, or dynamically by using the confighfs command. The buffer pool is logically partitioned into pageable storage and non-pageable storage, otherwise called permanently page-fixed storage. Actually, there is also a third category called "released" or "unbacked" storage which does not consume any real storage.

8.3.2.1 VIRTUAL parameter considerations

Virtual refers to the maximum amount of virtual storage which is to be used for the HFS buffer pool.

When the contents of a buffer are no longer needed, HFS must decide whether or not to release the real frames associated with the buffer pages. The VIRTUAL parameter is used to make that decision. If the virtual limit is currently exceeded, the buffer pages are released.

The advantage of releasing the storage is to reduce real storage usage. The disadvantage is the *Real Storage Manager* (RSM) overhead of acquiring new storage at the next time that the buffer is reused. VIRTUAL is not a hard limit on the size of the buffer pool, but is rather a target *working set size* of the HFS buffer pool.

Virtual is also used to determine how much data to cache per file. Whenever possible, up to 1MB of data is cached per every 2GB of VIRTUAL specified.

8.3.2.2 FIXED parameter considerations

FIXED is a target minimum number of megabytes of fixed buffers. The actual number can be (and often is) lower and may exceed the target if there is a lot of demand. These buffers remain permanently fixed regardless of HFS I/O activity. HFS may temporarily fix other buffers to meet the I/O demand since buffers involved in I/O must be fixed. In the performance statistics provided by RMF, the temporarily fixed buffers are not counted as "fixed"; The purpose of these statistics is to evaluate the effectiveness of permanently fixed buffers.

An optimal use of fixed buffers would avoid frequent page fixing, but would not waste storage. Wasting storage is only a problem if there is real storage contention. Assuming that there is a storage constraint, the frequency of HFS I/O, which can be determined from RMF's HFS report, is what determines the effectiveness of fixed storage. However, the definition of frequent is relative to other demands for storage. RMF computes a statistic called "%FIXED", which is the percentage of HFS I/O requests which did not have to dynamically fix a buffer. If %FIXED is 100, it means that there is ample HFS fixed storage, but the storage may be over-allocated. If it is believed that HFS I/O frequency is low, it may be desirable to reduce the FIXED value in order to free the storage for other use. If %FIXED is anything less than 100, then there is an opportunity to reduce the frequency of page fixing by increasing the FIXED value.

If a buffer needs to be acquired to perform an I/O operation and the buffer is not already fixed, then it has to be page-fixed to perform the I/O. If FIXED is non-zero, HFS tries to maintain a pool of permanently page-fixed storage so that it can avoid having to dynamically page-fix the storage. Page-fix operations are very costly in terms of CPU time, and even more costly in terms of the system serialization effects. At the present time, the RSM in OS/390 has one lock to serialize the allocation and page-fixing of real storage frames. Thus, by specifying a large FIXED parameter, CPU utilization may be reduced and overall system throughput may be increased. Such effects may be felt by both USS and non-USS applications.

However, there is a trade-off between the cost of fixing pages and the availability of free real storage. The danger in over-allocating fixed storage for the HFS buffer pool is that, if the buffer pool is under-utilized, the real storage is wasted and could be better used for other applications. In the worst case, if the fixed buffers are wasted and the system does not have enough real storage for other applications, the system may run out of real storage. In this case the system may be brought to a halt until the fixed buffers are freed. HFS receives a signal from

OS/390 when there is a real storage shortage and will attempt to free fixed buffers when this happens. In addition, HFS provides for the monitoring of its buffer pool usage, and it also provides the capability to dynamically modify the FIXED parameter using the confighfs shell command. Because of the potential for depleting real storage, system programmers should only set the FIXED parameter at a level that they can feel comfortable with. To set the FIXED parameter high may aggressively push the throughput to its limit, but it runs the risk of a real storage shortage when there is a shift in application usage away from HFS.

This is another downside of using fixed storage exists in an environment consisting of a mixture of USS and non-USS applications. For example, suppose that USS and DB2 coexist in the processor. Both applications use a large buffer pool to minimize I/O to the database. Before choosing a value for FIXED, the system programmer should choose the size of the buffer pool for each application. For HFS, the VIRTUAL parameter is used to control the size of the buffer pool. The system programmer's choice will depend on the relative importance of the USS and the non-USS applications.

8.3.2.3 Settings for Domino for S/390

When choosing an appropriate value for FIXED and VIRTUAL, it is useful to understand the dual purpose of HFS buffers. One purpose is to do I/O and the second purpose is to serve as a cache. Some applications do their own caching, in which case it is redundant for HFS to cache file data.

Domino for S/390 is such an application. Domino only uses HFS as an I/O driver and to cache the metadata about which Domino knows nothing. If the objective is to minimize I/Os, then it is desirable to use a large VIRTUAL parameter, while a small FIXED parameter is sufficient. As with any large buffer pool, specifying too large of a buffer pool may result in paging which is slower than I/O to HFS file systems. If it is expected that HFS is to be used primarily as an I/O driver, then it may be desirable to set FIXED equal to VIRTUAL, and to specify a value that is only large enough to efficiently handle the I/O.

8.3.2.4 Distributing the fixed storage among the four buffer pools

With respect to the management of permanently fixed storage, the 256KB buffer pool is treated differently from the other smaller pools because it is the largest buffer size, and because there is an opportunity to amortize the cost of the page fixing across many I/Os. Any sequentially accessed file larger than 64KB will use the 256KB buffer pool. A file that is larger than 768KB will be allocated four 256KB buffers, and will require one I/O to read each 256KB chunk of data. Unless VIRTUAL is larger than 2GB, subsequent I/Os will reuse the buffers.

For example, let us consider the case where we read a file that is 10MB in size. When the first megabyte is read, four 256KB buffers will be allocated and they will each be dynamically page fixed to perform four I/Os. If the system is not constrained by storage, HFS will not unfix the buffers when the I/Os complete. Since each buffer will be reused ten times to read the 10MB file, each page fixing operation is amortized across ten I/Os. When a 256KB I/O completes, if the current amount of fixed HFS storage exceeds the FIXED parameter, then the buffer is unfixed.

Given the nature of the amortization effects described above, HFS never permanently allocates fixed storage to the 256KB pool. If FIXED is specified in

the BPXPRMXX in PARMLIB, HFS automatically distributes the buffers to the three small buffer pools in a 4-2-1 ratio. If this distribution is not satisfactory, the confighfs shell command may be used to dynamically allocate fixed pages. If modified through the confighfs shell command, HFS allocates the fixed pages on demand. If the confighfs shell command is used to reset the fixed pages to zero, or to decrease the FIXED value, HFS will immediately attempt to unfix enough storage to meet the target.

If too many fixed pages should become allocated to a buffer pool that is no longer used, it is possible to provide HFS with an opportunity to redistribute the fixed pages among the pools by first resetting FIXED to zero, and then respecifying FIXED to some non-zero value. When FIXED is increased dynamically by the confighfs shell command and HFS is not used enough heavily to require more fixed buffers, then no additional buffers will be fixed and the total working set of fixed pages may be below the FIXED value.

8.3.3 fsync call considerations

If your programs or products issue an fsync call which sends a synchronous I/O request to the OMVS kernel, then all the other metadata pages in the same HFS data set will be synchronously hardened. This implementation is done because of the index shadowing which enables HFS file systems to maintain their data integrity when the system crashes.

If the specific files in one HFS file system are updated frequently by a program using fsync calls, you should separate those files into other HFS file systems to avoid the influences on other files.

There is no way to monitor or report on fsync calls. If fsync calls are a concern, you must check with the appropriate application developers.

8.4 Sysplex and HFS sharing

There are a number of performance implications that you need to be aware of if you share HFS files between OS/390 systems in a sysplex.

8.4.1 XCF overhead

The intersystem communication required to provide additional availability and recoverability of shared HFS affects response time and throughput on file systems being shared in a sysplex. Increased XCF message traffic to support shared HFS can contribute to system degradation if not monitored and controlled. Workloads that use large file buffer sizes will give you an increased number of large sized messages; Workloads with small file buffer sizes will you give you an increased number of small sized messages. To control XCF message traffic, closely monitor the number and size of message buffers and the number of message paths within the sysplex. It is likely that you will need to define additional XCF paths and increase the number of XCF message buffers above the minimum default.

For more information on signaling services and determining message buffer sizes, see *OS/390 V2R9.0 MVS Setting Up a Sysplex,* GC28-1779.

Additional information is also available in the Parallel Sysplex Test Report, on the OS/390 UNIX System Services web site at:

`http://www.s390.ibm.com/unix`

Also see the Technical Support Information Site, *"Parallel Sysplex Performance: XCF Performance Considerations", Flash 10011* which can be viewed at:

`http://www.ibm.com/support/techdocs`

8.4.2 Locally mounted versus remotely mounted HFS

In many cases, when accessing data on a system that owns a mounted file system, the file I/O time is only the path length to the buffer manager to get the data from cache. However, file I/O to a shared file system from a client that does not own the mount has additional path length plus latency involved in the XCF messaging function. Under some circumstances it is reasonable for it to take much longer for remote file access compared to locally mounted file access. For this reason, we recommend that when you mount an HFS file, mount it on that system where it is most heavily used.

Example test configurations and response times can be found on the OS/390 UNIX System Services Web site:

`http://www.s390.ibm.com/oe/`

8.4.3 How shared HFS affects mount times

You should be aware that because of I/O operations to the mount table CDS (Couple Data Set), every mount request requires additional system overhead. Mount time increases as a function of these three parameters:

- Number of mounts
- Number of members in the sysplex
- The size of your CDS

8.4.3.1 Number of mounts

When a system joins the sysplex, information concerning its mounts must be written into the CDS; this takes time since mounts are performed serially, one at a time. Also, every time a system joins the sysplex, it must first read the CDS and perform the mounts already listed there. Once this is done, the new system can perform its own mounts and write that information to the CDS. As each new mount is performed, if there are other systems in the sysplex, these other systems must then read what was just written into the CDS and perform the same mount (catch-up). Conducting these catch-ups affects the system on which the new mounts are initiated, that is, the next mount can not be written into the CDS until the other systems have read and mounted the last mount recorded in the CDS. Thus, the more mounts that need to be performed, the more time is expended in the reading of and writing to the CDS.

8.4.3.2 Number of members in the sysplex

As pointed out above, when a new system enters a sysplex, it must perform all the mounts listed in the CDS before it can perform its own mounts. The more systems in the sysplex, the more mounts there will be recorded in the CDS, and, therefore, the more time it will take for this new system to read what is already in the CDS and perform the mounts. Also, as already discussed, a catch-up function must be performed when there are multiple systems in the sysplex. The more

systems in the sysplex, the more systems that will be competing to read what is added to the CDS and perform the new mounts listed there. This competition, as discussed above, impedes the mount response time on the system that is writing new mounts to the CDS.

8.4.3.3 Size of the CDS

When you format your CDS, the number of mounts specified in the JCL determines the size of the CDS. A value of 1000 gives you a smaller CDS than a value of 5000. When a system reads the CDS for mount information, it must read the entire CDS. Therefore, even if you are only performing 600 mounts, if you specify a mount number of 1000, all members in the sysplex will read the CDS as if 1000 mounts must be performed. In other words, whatever amount of space the CDS will use to hold information on 1000 mounts will be read by all the systems in the sysplex.

For example, even if only 600 mounts are listed in the CDS, if the mount number in the JCL is 1000, a new system joining the sysplex will read the entire CDS, not just the space that contains the information on the 600 mounts. Also, those systems performing catch-up will read not only the new information added, but will first read through the entire CDS to get to the new information along with whatever additional space is left over (in our example, the space used up by the 400 additional mounts defined in the JCL).

Important Information

We recommend that you select an appropriate number of mounts to be specified in the JCL; This way you will not end up with a CDS whose size is unnecessarily large.

More information on defining and formatting the CDS can be found in the following publications:

- *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890
- *OS/390 V2R9.0 MVS Setting Up a Sysplex,* GC28-1779

Example test configurations and mount response times can be found on the OS/390 UNIX System Services Web site:

`http://www.s390.ibm.com/unix`

8.5 Distributed File System considerations

A file system can only be exported by the Distributed File System (DFS) server at the system that owns the file system. Once a file system has been exported by DFS, it cannot be moved until it has been unexported by DFS. To recover from system outages, you need to weigh sysplex availability against availability to the DFS and Server Message Block (SMB) clients. When an owning system recycles and a DFS-exported file system has been taken over by one of the other systems, DFS will not be able to automatically re-export that file system. The file system will have to be moved from its current owner back to the original DFS system, the one that has just been recycled, and then re-exported. For file systems that are mostly for use by DFS clients, you should consider specifying NOAUTOMOVE on the MOUNT statement so that they will not be taken over if the system is recycled, and they will be available for automatic re-export by DFS.

8.6 Using the copytree utility

Over a period of time, as more and more updates are done to an HFS file, it can get fragmented. You can use the Copytree utility to 'defrag' the HFS file. This can improve the performance of the HFS file. For more details of the Copytree utility please refer to: 5.3.1, "Copytree utility" on page 127; and 6.4.2, "Reorganizing HFS for performance" on page 165.

Chapter 9. Implementing HFS for selected applications

This chapter discusses the HFS data set allocations and the backup policies for selected applications on UNIX System Services. The selected application include:

- UNIX System Services on OS/390 2.7 and 2.9
- Domino for S/390
- WebSphere for OS/390
- OS/390 Network File System Server

9.1 Understanding your backup needs

Before discussing each application from a backup policy standpoint, we first categorize the characteristics of the data on the HFS file systems.

In general, the characteristics of the data determine the recovery needs, which, in turn, impose the backup requirements. The following examples describe some possible scenarios in USS environments.

Discardable data: The customer may find the risks or consequences of losing a particular type of data acceptable. However, this is not a workable plan in a disaster recovery situation, where many or all files must be rebuilt. To avoid recreating a large numbers of files, it might be simpler to take low-frequency data set dumps, using a tool such as DFSMSdss or DFSMShsm.

Static data: The root file system of USS, which contains many binary executables, is an example of data which is very static. At first glance, it might seem that a post-installation one-time backup would suffice for later recovery. However, the problem is that the content of the directory is not totally static, because there is a possibility for certain processes updating a file. Since these updates are not well documented, it is advisable to base the recovery strategy upon full periodic backups. Since this data is contained in a single HFS, DFSMSdss logical data set dump or DFSMShsm incremental backup might be the appropriate tools.

Non-critical data: Data is non-critical when business needs make it acceptable to recover the data to a given point in time in the not too distant past. This requirement could be accomplished through periodic backups by DFSMSdss data set dump or DFSMShsm incremental backup, and then a restore of the last backup when needed. Any updates that may have taken place after the backup would be lost, but this is acceptable by definition.

Critical data: This category involves files where the business need dictates that the data must be recovered to its state immediately preceding the failure. The only way to accomplish this type of recovery is with periodic backups combined with some journal-based forward recovery system. The new transaction log in Domino for S/390 Release 5 allows such solutions.

The categories above are applicable to the backup policies discussed in the following sections.

9.2 UNIX System Services (OS/390 2.8 and before)

When OS/390 2.7 is installed according to a ServerPac scenario, the following HFS structure is built as shown in Figure 98.

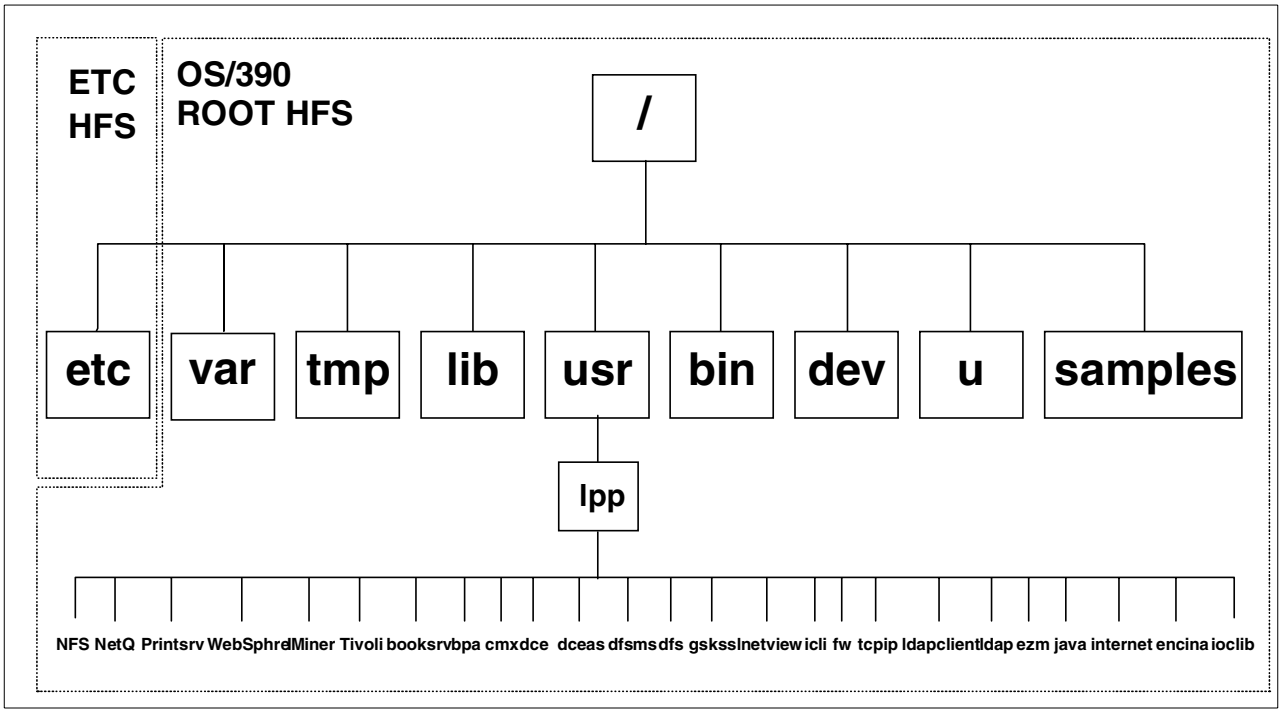


Figure 98. HFS structure with OS/390 2.7

After the installation, there are two separate HFS data sets created, one under the root (/) and the other under the /etc directory, and there are many files and executables of various products under the /usr/lpp directory.

9.2.1 Recommended HFS structure and management (UNIX)

First of all, we recommend that you separate volumes (or storage groups) containing HFS data sets from volumes holding other OS/390 data sets whenever possible.

In addition, we recommend that you separate HFS data sets for the mount points /var, /tmp and /u, because they each have unique characteristics regarding their usage. Following are some data management considerations for these HFS data sets.

9.2.1.1 Root file system

The root file system is an aggregation of read-intensive files, such as LINKLIB on OS/390. If we lose the root file system, USS does not work.

We recommend making a copy of the root file system, using another data set name. This copied data set can then be used for the product maintenance activities. If the root file system is damaged, you can use this backup.

The root file system is considered **static data**.

9.2.1.2 /etc directory

This directory contains important customized data, which is used, for example, by USS or other product initialization processes.

This file system is considered **non-critical data**.

9.2.1.3 /var directory

This directory contains dynamic data used internally by products and by elements and features of OS/390. Any files or directories needed are created during the execution of code. An example of this is caching data. IBM products will only create directories or files when code is executed.

The typical process using /var is the *syslogd daemon*, which writes various log records from USS and TCP/IP applications to files under the /var directory, or sends those records to other systems through the network. Basically, these log record files continue to increase, so a space management task to delete increasing HFS files can be performed automatically by the *cron daemon*, which can periodically run a shell script to delete files.

This file system is considered **discardable data**. If these log records are used as statistics data, this file system will be considered **non-critical data**.

9.2.1.4 /tmp directory

You should use the option of mounting a Temporary File System (TFS) on /tmp rather than using an HFS file system for it, because this directory contains temporary data used by products and applications such as the sort program. TFS enables you to get better performance. For more information about TFS, refer to *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890.

9.2.1.5 /u directory

This directory contains individual USS users' HFS data sets, and almost all USS users have their home directory under the /u directory. We recommend that you use the *Automount Facility* under the /u directory. The advantage of the Automount Facility is as follows:

- With automount active and the correct automount policy in place, there is no need to create a user directory with the mkdir command; the user directory will be dynamically allocated and the user's HFS data set will be automatically mounted at the */u/userid* mount point. Later, if the user's file system has not been accessed, based on certain time criteria in your automount policy the user's HFS data set will automatically be unmounted.
- Another advantage of using the Automount Facility is to manage user file systems (as opposed to adding them to BPXPRMxx). The last access time of the data set is updated only when the data set is mounted automatically. If a user goes on vacation for two weeks, then their HFS might not be mounted, and therefore can become a candidate for DFSMShsm migration. If the user HFS data sets are defined in the BPXPRMxx, the HFS data sets will always be mounted, and so the last access date will always be updated.

For more information about the *Automount Facility*, refer to A.2, "Automount facility" on page 248.

Basically, these file systems are considered **non-critical data**.

These data sets have similar characteristics to users' TSO data sets. So, if you already have backup or migration policies for TSO data sets, you can adopt them for these HFS data sets.

9.2.2 Recommended HFS allocations (UNIX)

We recommend the following data set allocations on physical volumes (storage groups) as shown in Figure 99.

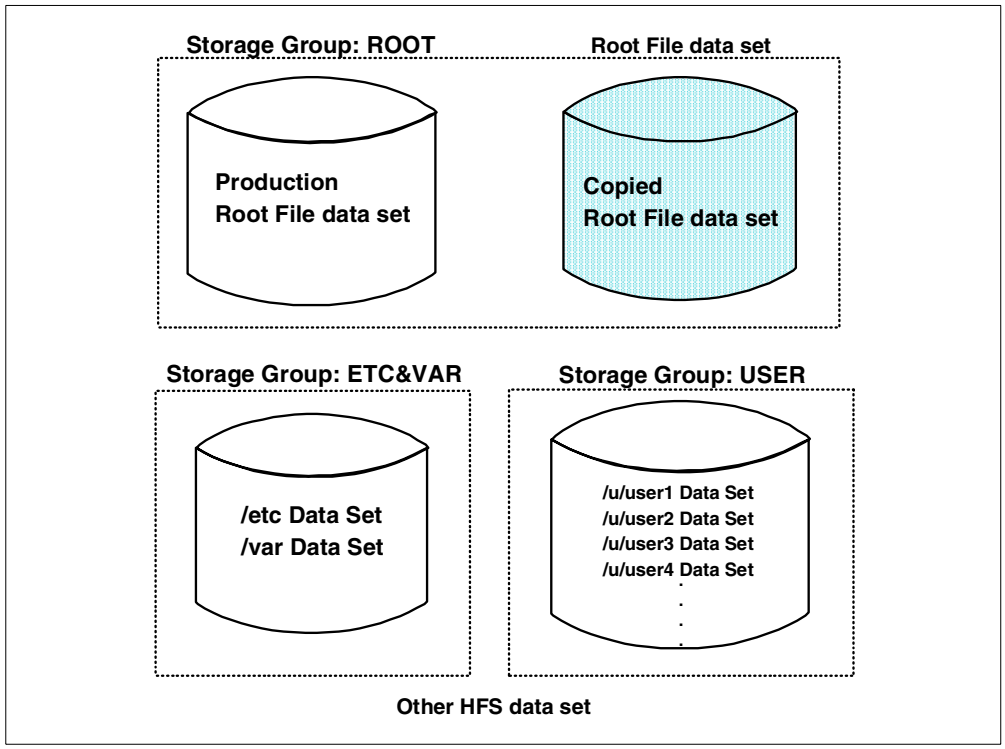


Figure 99. Recommended UNIX System Services HFS allocations

Considerations for each data set are summarized as follows:

- **Root file data set:** The production and copied data set can be allocated on the same storage group, but should be allocated on different volumes.
- **Other HFS data sets:** These should ideally be divided into two storage groups, one group for /etc and /var, and one group for /u.

  If you have few USS users, you can use the same storage group for /etc, /var, and /u.

9.3 UNIX System Services (OS/390 2.9 and later)

OS/390 2.9 along with DFSMS 1.5 now support sharing of HFS data sets between multiple OS/390 systems that are members of the same sysplex. When you install applications to be run under OS/390 Unix System, you now need to consider the implications and think about how you can exploit this new facility.

Although, we recommend that you exploit shared HFS support, you are not required to. If you choose not to, you will continue to share HFS data sets as you have before OS/390 2.9.

Before OS/390 2.9, users logged onto one system in a sysplex had write access only to the file systems associated with their own system. For example, the system programmer who makes configuration changes for the sysplex must log onto each system and change entries in the /etc file for that system. With shared HFS, The system programmer can logon to any system and then can change entries in all /etc file systems. The changes will be visible from all systems.

Also with shared HFS, there is greater availability of data in case of system outage.There is also greater flexibility for data placement.

You now need to understand the new HFS files introduced for sysplex sharing. These are

- Sysplex Root
- Version HFS
- System Specific HFS

For an explanation of these, please refer to Chapter 10, “HFS sysplex sharing implementation” on page 221.

9.4 Domino for S/390

Important information

To get the latest Domino for S/390 R5 implementation, see also:

- *Lotus Domino for S/390 Problem Determination Guide*, SG24-5599
- *Lotus Domino for S/390 Release 5: Installation, Customization and Administration*, SG24-2083

When Domino for S/390 is installed according to the installation procedures, the three separate HFS data sets shown in Figure 100 are built, with the logical file structure shown in Figure 101.

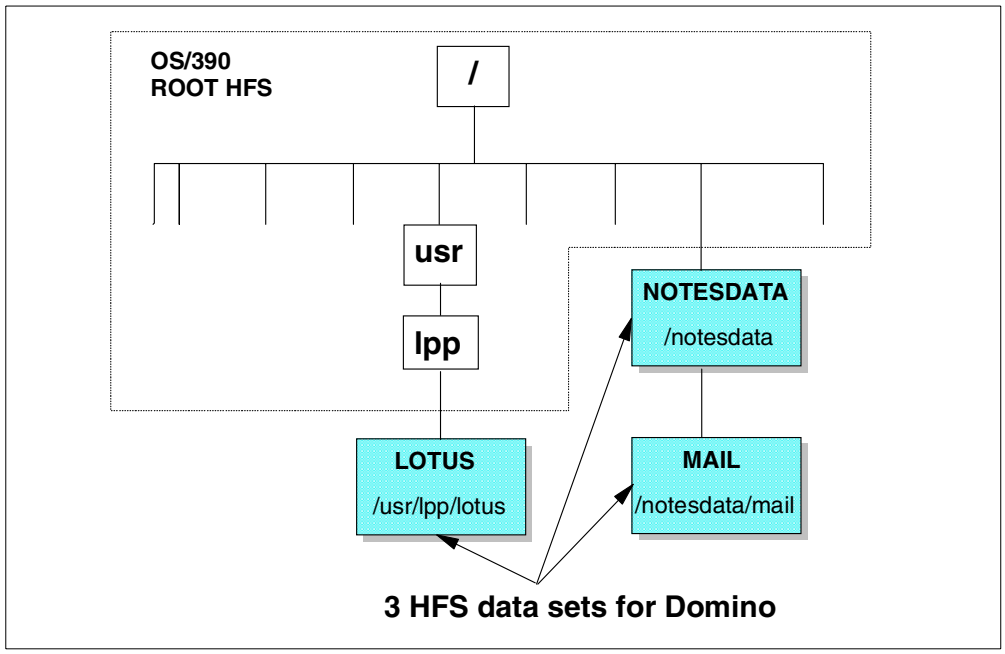


Figure 100. HFS data sets structure of Domino for S/390 installation

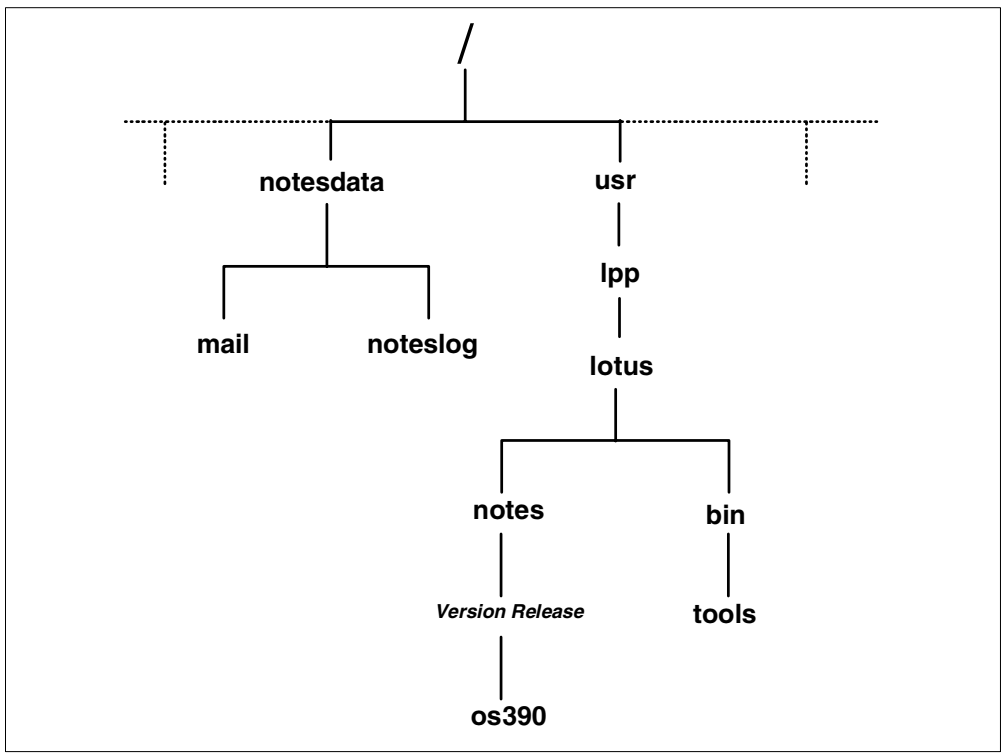


Figure 101. Logical file structure for Domino for S/390

9.4.1 Recommended HFS structure and management (Domino)

The recommendations for using a Domino HFS structure are:

- One more HFS data set should be allocated on the mount point /notesdata/noteslog, because files under /notesdata/noteslog tend to increase in size. If these log records make the HFS data set full, other files under /notesdata cannot be updated, and this can stop the Domino server activities.
- In general, the Mail HFS data set should be separated into several data sets, and the directory structures should be changed according to the recommendations in Domino for S/390. Methods for separating this file system are not discussed in this redbook, but following are some recommendations from a storage management standpoint.

9.4.1.1 Lotus HFS data set

The Lotus HFS data set, which is mounted on /user/lpp/lotus, consists of several executables and shell scripts related to the Domino server function, and some tools. This data set is an aggregation of read-intensive files, but it is not totally static; For example, some updates take place when a new server partition is added. If we lose this data set, the Domino server will not run.

We recommend making a copy of the Lotus HFS data set, using another data set name. If the Lotus HFS data set is damaged, you can use this backup copy.

This file system is considered **static data**.

9.4.1.2 Notesdata HFS data set

The Notesdata HFS data set, which is mounted on /notesdata, has various important files used to initialize and manage the entire Domino server. Some of them are updated dynamically.

This file system is considered **non-critical data**.

9.4.1.3 Noteslog HFS data set

The Noteslog HFS data set, which is mounted on /notesdata/noteslog, consists of various log files. These log files continue to increase in size, so the space management to delete the increasing files can be performed automatically by the *cron daemon*, which can periodically run a shell script to delete files.

This file system is considered **discardable data**. If these log records are used as statistics data, this file system is considered **non-critical data**.

9.4.1.4 Mail HFS data set

The Mail HFS data set has the following characteristics:

- In general, this HFS data set would be separated into several data sets after the Domino installation.
- This HFS data set consists of many Notes client databases (files, from an HFS standpoint), and they will sometimes need a large amount of storage space.
- Most update I/Os to the Domino system are concentrated in this data set.
- This file system can be considered **critical** or **non-critical**, depending on your requirements.

You can choose one or a combination of the following methods to recover a Mail HFS data set or each Notes database.

- Notes-based database replication

- Tivoli Storage Management (former ADSM) to back up each Notes database in the HFS data set
- USS archive commands to back up each Notes database in the HFS data set
- DFSMShsm to back up the HFS data sets
- DFSMSdss to back up the HFS data sets
- RVA based backup options (Snapshot)

Most importantly, when you restore a backed-up HFS data set or database, you need to decide how to recover it to the latest date. Methods for recovering Notes databases of *critical* data are not discussed in this topic. For more detailed information about the Notes database recovery of *critical* data, refer to the appropriate Notes publications.

Following are some considerations for using DFSMSdss and DFSMShsm to back up HFS data sets:

The DFSMSdss quiesce process for mounted HFS data sets allows backups to be taken while the owning Domino server is active. In this scenario, the file system can be quiesced between writes, in the middle of a database update by the Domino server. The resulting backup would have a database with incomplete data. What would happen if the need to restore arose, and the Domino server was started with the restored database? The startup database consistency check scan would detect the partial update, and a subsequent FIXUP would remove it. This is acceptable if the partial update was the only update performed by the application's logical unit of work. In addition, it is obvious that all other successful updates from the time of the backup would also be lost.

Another scenario would be when the quiescing and backup may take place at a clean point, meaning that there are no documents being updated; But, if the application's logical unit of work consists of updates to two or more documents, the server may be caught in between updates during the backup. Domino has no externalized concept of a logical unit of work, nor the corresponding controls for it, such as syncpoint, bucket, and so on. Under these conditions, the backup would contain logically incomplete data, although not inconsistent from a database structure point of view. This is a serious issue that must be considered.

In addition to the previous considerations, the time involved in dumping large HFS files can translate into unacceptable system quiesced times.

Since DFSMShsm uses DFSMSdss as the data mover, its use has issues similar to the use of DFSMSdss. In addition, you must consider the timing of the backup in relation to the status of the Domino server. This is particularly critical if the backup is to be taken with the server down.

9.4.2 Recommended HFS allocations (Domino)

The recommendations for data set allocation on physical volumes (or storage groups) are shown in Figure 102.

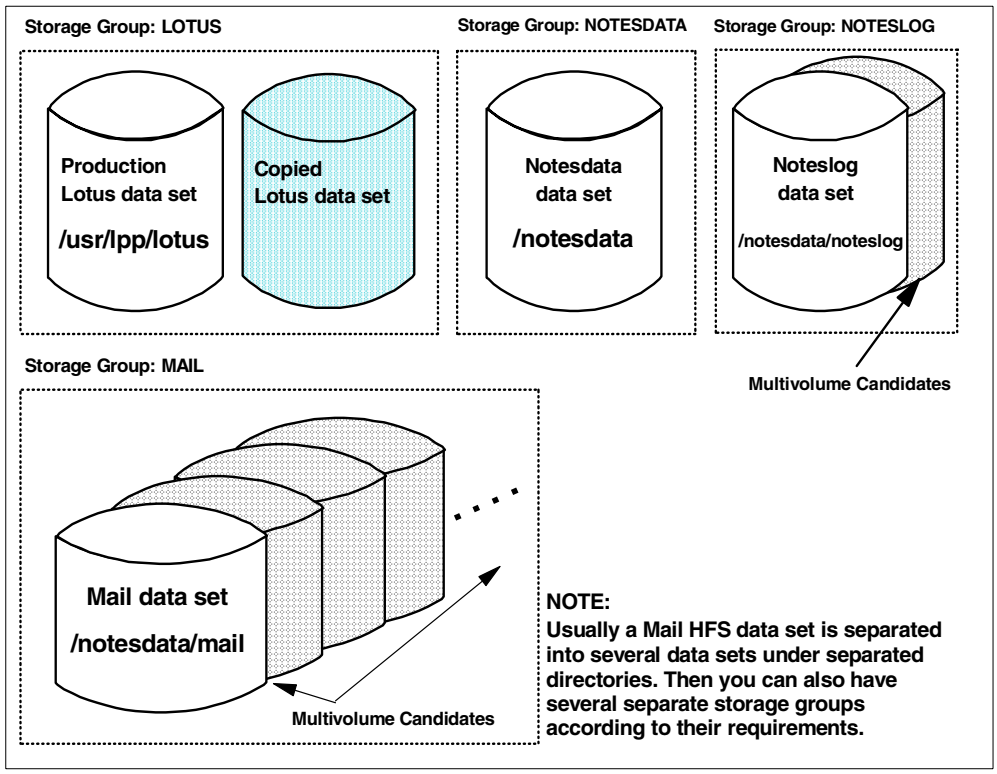


Figure 102. Recommended Domino for S/390 HFS allocations

Some considerations for each data set are:

- **Lotus HFS data set:** The production and copied data set can be allocated on the same storage group, but should be allocated on different volumes.

  These data sets can be allocated on the USS storage group ROOT, but you should avoid allocating the production Lotus data set on the USS production root volume.

- **Notesdata HFS data set:** The Notesdata data set should be allocated in an independent storage group or a volume which is not influenced by other I/O activities.

- **Noteslog HFS data set:** The Noteslog data set should ideally be allocated in an independent storage group for performance reasons.

  From DFSMS 1.5 on, this data set can be considered a multi-volume allocation candidate.

- **Mail HFS data sets:** The Mail HFS data set would usually be split into several data sets under separate directories, and then, depending on the disk subsystem used, several separate volumes would be used.

  From DFSMS/MVS 1.5 on, this data set can be considered a multi-volume allocation candidate, both to avoid running out of space and to support large capacities for Notes client databases.

9.5 WebSphere for OS/390 case

When WebSphere for OS/390 is installed according to the installation procedure, the following HFS data sets are created and shown in Figure 103.

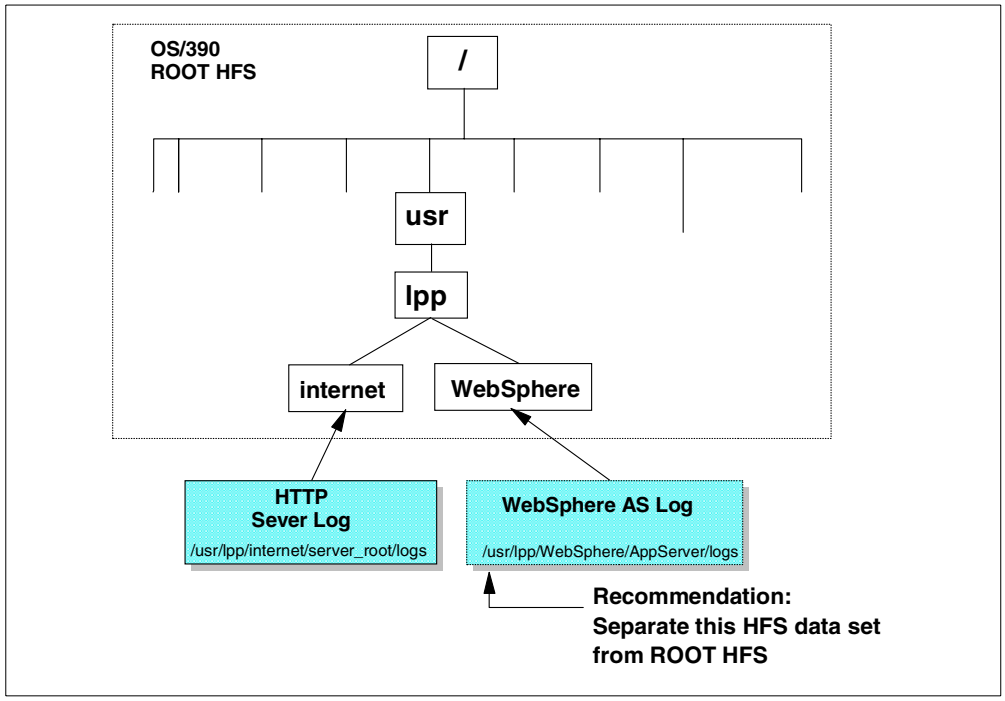


Figure 103. HFS data sets structure of WebSphere for OS/390

We recommend you have one HFS data set for the http server logging and another HFS data set for the WebSphere application server logging.

9.5.1 Recommended HFS structure and management (WebSphere)

We recommend that you have a separate HFS data set on the mount point /usr/lpp/WebSphere/AppServer/logs. Following are considerations for data management of these HFS data sets.

9.5.1.1 Logging file system

The HFS data sets, which are mounted on /usr/lpp/internet/server_root/logs and /usr/lpp/WebSphere/AppServer/logs, consist of various logging files. They are not used for the transaction recovery like other transaction subsystems. Rather, they are used to report the system activities and the error records.

These log files continue to increase in size, so space management to delete increasing files can be performed automatically by the archive function of WebSphere.

These file systems are considered **discardable data**. If you use these log records as statistics data, this file system is considered **non-critical data**.

If these HFS data sets are full, the Web server will slow down and use up all available CPU. Furthermore, you might not even be able to start your Web server. You should pay close attention to the usage of these data sets.

9.6 OS/390 Network File System Server

The OS/390 Network File System (NFS) Server provides a way for client machines to access data held on an OS/390 system. The latest level of the NFS server was shipped with OS/390 2.6 and added support for NFS version 3 protocols, WebNFS protocols and advisory locking using the Network Lock Manager and Network Status Manager. The NFS server provides file access across an IP network and can serve data from these type of OS/390 data sets:

- Sequential (including extended format)
- Partitioned (both PDS and PDSE)
- VSAM KSDS, ESDS or RRDS
- Direct access (DOSRG=DA)
- UNIX file systems held in HFS data sets

Any of these data sets may be system-managed or not. For anything other than HFS, it is better to use system-managed data sets for performance reasons. For HFS data sets, your storage management policies can dictate whether they should be SMS managed or not.

Normally, an OS/390 data set appears to the NFS client as a file. The exceptions are for PDSs and PDSEs where the data set name appears to be a directory name and the individual members appear as files within that directory.

The relationship between the clients, the OS/390 NFS Server and the data is shown in Figure 104.

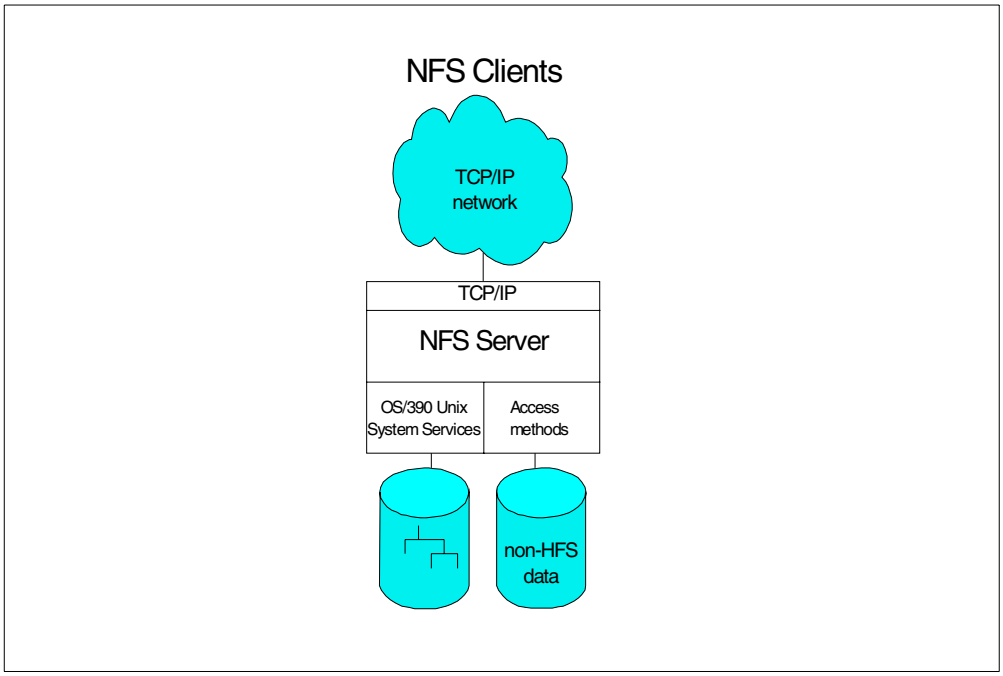


Figure 104. Where the OS/390 NFS server fits

You may choose to use the NFS server for different reasons. You may want to give workstation users direct access to data held on the OS/390 system or you may want to use the OS/390 system as a file server.

If you want to use the OS/390 system as a file server, storing data in the HFS has a number of advantages when compared to using other OS/390 data sets:

- You can use UNIX style names with file names up to 255 characters and path names up to 1023 characters.
- Mixed-case file names and file names with special characters (except nulls, slash and commas) are permitted.
- You can use a hierarchical directory structure.
- Access control uses UNIX-style permissions, with group and user id support at file level.

It has sometimes proven difficult to implement application packages with data stored on OS/390 and accessed through the NFS server because the application may be written to need a file name format that is not a valid OS/390 data set name. Serving data from an HFS will avoid that problem.

If the NFS server is used to provide file sharing for workstation users, it is possible that they will understand file naming rules for their workstation but will not understand OS/390 data set name rules. Using HFS data sets is much easier for them.

In addition, if you use non-SMS OS/390 data sets, you may see poor performance when the client requests file size information as the entire data set must be read to determine the size.

For anything other than an HFS file system, the time and date stamps do not have exactly the same meaning as a client file. For HFS, the meanings are identical.

You may request synchronous or asynchronous processing on the **mount** command when you use NFS version 2 protocols. Asynchronous processing allows the data to be written to disk after the server responds that a write request has been completed.

In order to access HFS files, you need to know the HFS prefix that has been defined. You can find this out using the **showattr** command. The default is /hfs. So, to mount the HFS directory /james from OS/390 host sc64 on an AIX system at mount point /u/james/mnt, this command should be issued from AIX:

```
mount sc64:"/hfs/james" /u/james/mnt
```

For more information on NFS facilities available within OS/390, please refer to *OS/390 Network File System User's Guide,* SC26-7254.

9.7 Other applications

If you run other USS applications using HFS data sets, you must know the characteristics of their data (criticality, frequency of changes, and so on) in order to determine the storage management needs. Then, you can adapt them to one of the backup categories discussed in 9.1, "Understanding your backup needs" on page 209 which will help you to decide upon an appropriate storage management policy.

Chapter 10. HFS sysplex sharing implementation

The main features of HFS sysplex sharing are:

- A common file system hierarchy on all systems in the sysplex
- The ability to write to file systems from all systems in the sysplex
- One common BPXPRMxx member for all systems in the sysplex
- Sysplex wide mount requests

In this chapter first we explain how HFS sysplex sharing works. Then we describe how to implement it step by step.

10.1 How HFS sysplex sharing works

Every HFS data set that has shared read/write in the sysplex, has a central owning system that manages it on behalf of other systems in the sysplex. The owning system gets read or write requests from other systems and performs these operations on the HFS data sets. A messaging protocol via XCF services is used to transfer data around the sysplex from the central owner. This is like a client and server environment. For more details, please see Chapter 7, “Sharing and serialization for HFS data sets” on page 167.

When HFS data sets are shared across the sysplex, you will have a big pool of all the HFS data sets of the entire sysplex. All these HFS data sets will be accessible to every system in the sysplex. This raises a number of questions.

- The systems in the sysplex could be at different release levels. You will have different versions of root HFS in the system and every system in the sysplex will have access to all these root HFS files. How does each system pick up the right version of the root HFS?
- You have HFS files and directories specific to each system in the sysplex like /etc, /var, /dev and /tmp. All the systems in the sysplex will be able to see the files of every system. How does each system point to the correct HFS files specific to that system?
- What happens to the HFS files of a particular system when it joins the sysplex or gets out of the sysplex?
- What if I want to upgrade to OS/390 2.9, but do not want HFS sysplex sharing?

This section answers these questions by explaining how the new HFS file system structure (that was introduced in OS/390 2.9) meets these requirements. You will also see how the same structure accommodates sysplex and non-sysplex environments.

The file structure has changed and looks different to the systems programmer. But to each system in the sysplex, the file structure looks just like the structure before OS/390 2.9.

10.1.1 Sysplex root

This is a new structure that was introduced in OS/390 2.9 to support HFS sysplex sharing. The sysplex root is an umbrella structure to encompass all the HFS files in the entire sysplex. It contains directories and symlinks to redirect HFS file

references to the correct files across the sysplex. It does not contain any files or any code. The sysplex root is created using JCL supplied in SYS1.SAMPLIB. Figure 105 shows the structure of the sysplex root after it has been created.

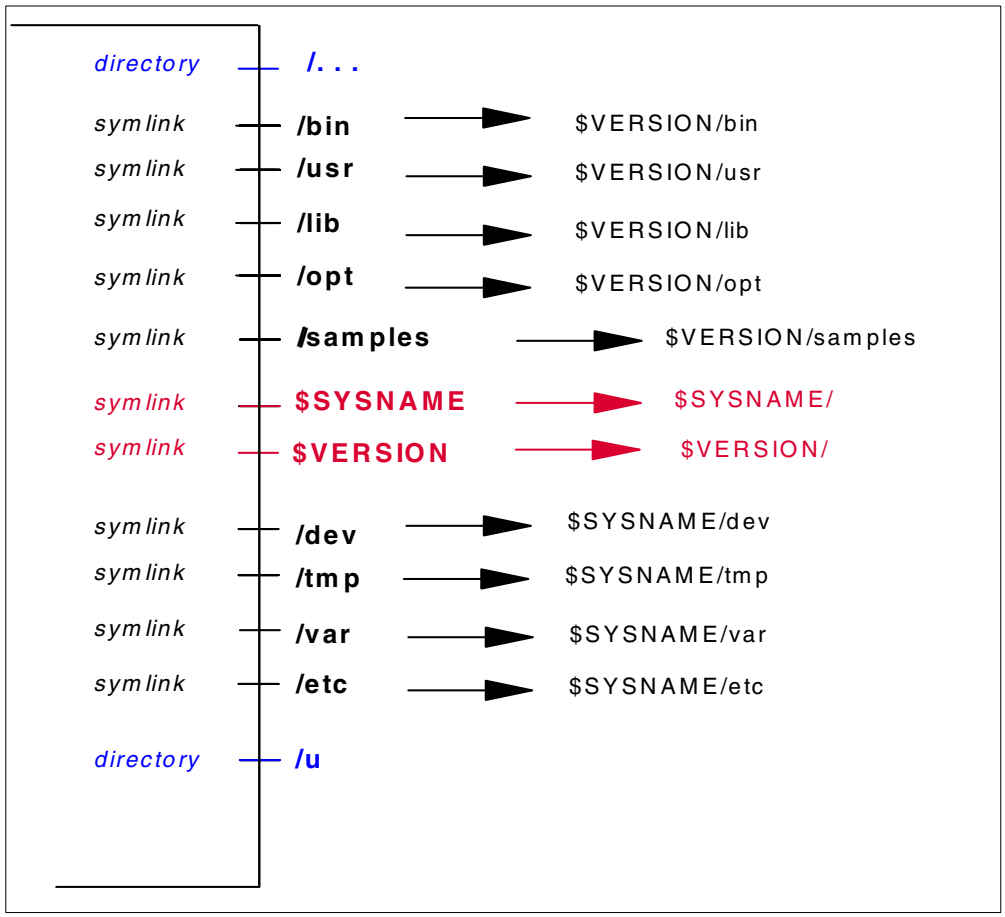


Figure 105. Sysplex root structure

Please note that the sysplex root has symlinks to point to /bin, /usr/, /lib and /opt files that are specific to a release/service level of OS/390 and /dev, /tmp, /var, and /etc files that are specific to each system in the sysplex.

10.1.2 Version HFS

The version HFS is the IBM supplied root HFS data set containing files and executables for OS/390 elements. To avoid confusion with the sysplex root HFS data set, the IBM supplied root HFS data set is hereafter called Version HFS. In a sysplex environment, there could be a number of Version HFS data sets, each denoting a different release or service level of OS/390.

10.1.2.1 Structure of Version HFS

Please see Figure 106. Note that the version specific files /bin, /usr, /lib, /opt and /samples have symlinks.

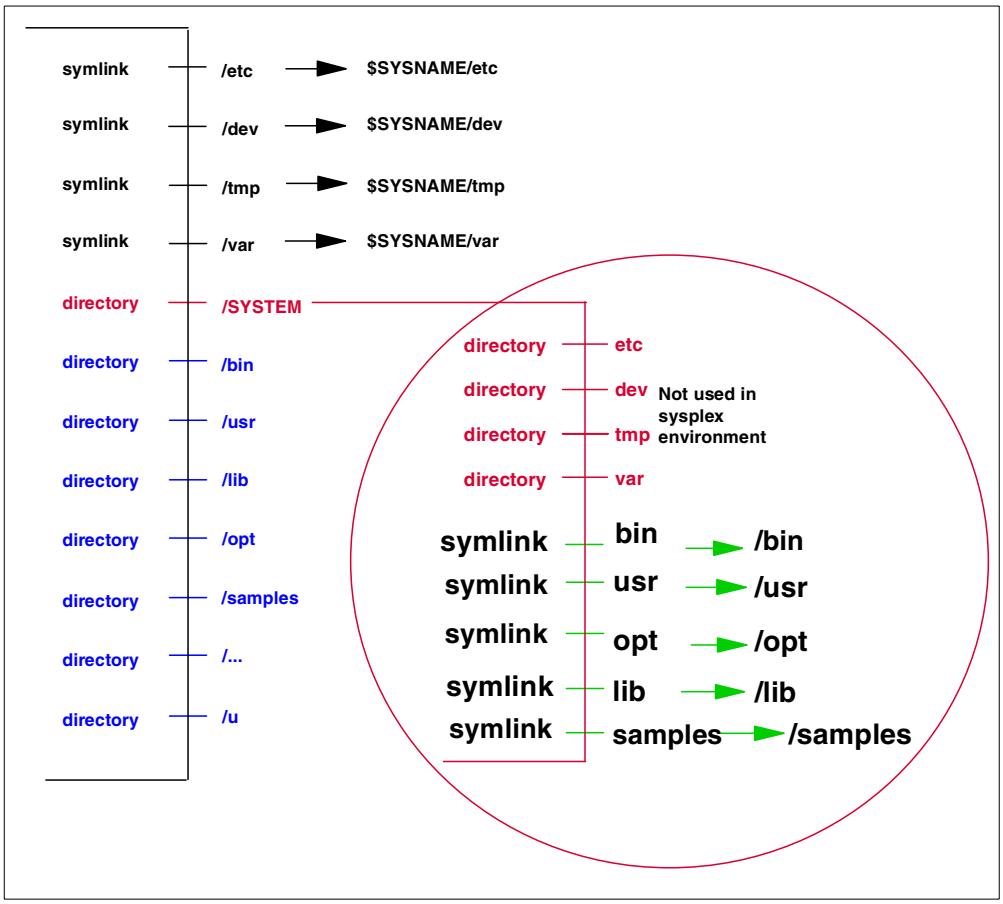


Figure 106. Structure of Version HFS

10.1.2.2 Setup of Version HFS

In the sysplex environment each system could be on a different OS/390 release or service level. In this section, we describe how the sysplex root is set up to point to the correct Version HFS for any system in the sysplex.

You use the VERSION statement of BPXPRMxx member to specify the version of the HFS. Figure 107 shows how to set up the Version HFS using the BPXPRMxx parmlib member.

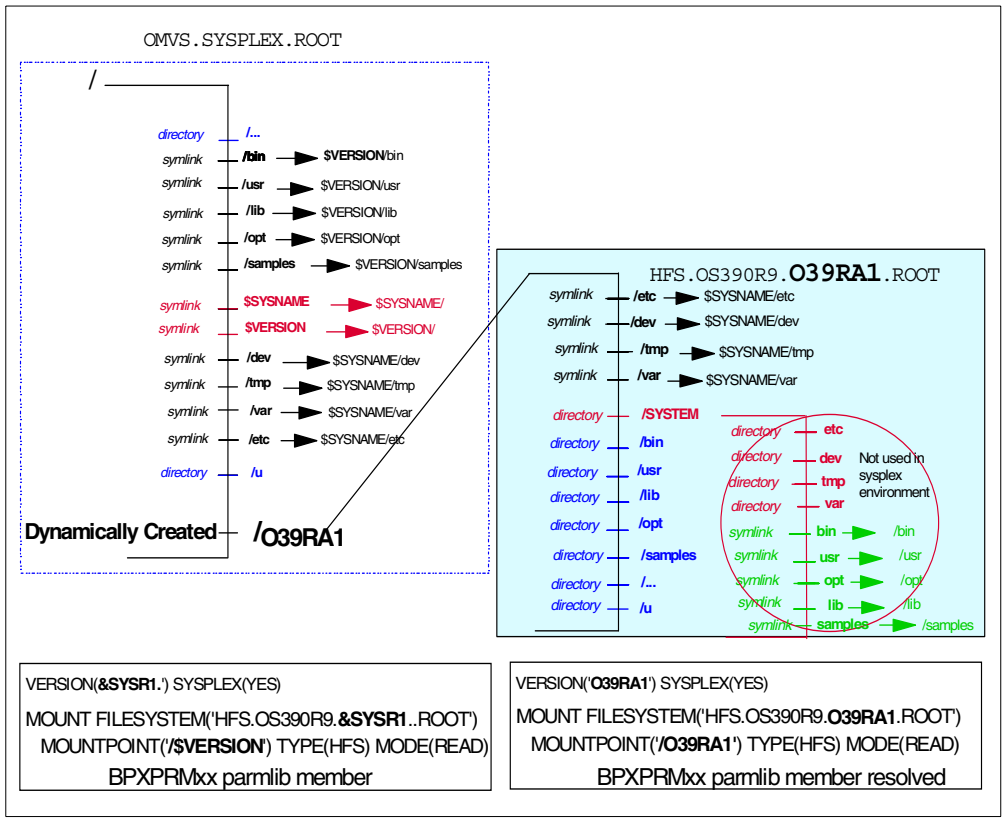


Figure 107. Set up of the Version HFS

In this example we used the system residence volser (symbol **&SYSR1**) to identify the Version HFS. Also note that we use the same **&SYSR1** as one of the qualifiers of the version HFS data set. Our system residence volume was **O39RA1**. Here is what happens when the system processes the BPXPRMxx member at the time of IPL.

The statement:

```
VERSION(&SYSR1.) SYSPLEX(YES)
```

Which gets resolved to:

```
VERSION(O39RA1) SYSPLEX(YES)
```

The SYSPLEX(**YES**) parameter tells the system that we want to do HFS sysplex sharing. The system creates a mount point **/O39RA1** in the sysplex root and also sets the symlink **$VERSION** to **/O39RA1**

The mount statement:

```
FILESYTEM('HFS.OS390R9.&SYSR1.ROOT)' MOUNTPOINT(/&VERSION)
```

Which gets resolved to:

```
FILESYTEM('HFS.OS390R9.O39RA1.ROOT)' MOUNTPOINT(/O39RA1)
```

This will mount the HFS Version data set on mount point **O39RA1**.

After IPL when the system tries to access HFS files /bin, /usr, /lib or /opt, the symlinks resolve them to /**O39RA1**/usr, /**O39RA1**/usr, /**O39RA1**/lib and /**O39RA1**/opt. The mount point /**O39RA1** will then have the right Version HFS mounted and these references will pick up the right HFS files.

10.1.3 System specific HFS

Files like /dev, /tmp, /var and /etc are specific to each system. In this section we show how the system uses symlinks to point to the correct system specific HFS files for each system.

You have to define a root HFS and /dev, var, /tmp and /etc HFS files for each system. We recommend that you use **&SYSNAME** as one of the qualifiers of these files to identify the system to which these files belong to.

Please see Figure 108 on page 226 to understand how the symlinks are resolved to refer to the correct HFS files for each system.

In this example we used system **SC64** for **&SYSNAME**. At the time of the IPL the system reads the BPXPRMxx member. The parameter SYSPLEX(**YES**) tells the system that we are going to do HFS sysplex sharing. The system dynamically creates a mount point **SC64** on the sysplex root. Then it processes the BPXPRMxx statement:

```
MOUNT FILESYSTEM('WTSCPLX2.&SYSNAME..SYSTEM.HFS')
   MOUNTPOINT('/&SYSNAME.') NOAUTOMOVE TYPE(HFS)  MODE(RDWR)
```

Which gets resolved to:

```
MOUNT FILESYSTEM('WTSCPLX2.SC64..SYSTEM.HFS')
   MOUNTPOINT('/SC64') NOAUTOMOVE TYPE(HFS)  MODE(RDWR)
```

And, the system specific root for system **SC64** gets mounted on mount point **SC64** of the sysplex root.

Now the system processes the BPXPRMxx statement:

```
MOUNT FILESYSTEM('HFS.&SYSNAME..DEV') MOUNTPOINT('/&SYSNAME./dev')
NOAUTOMOVE TYPE(HFS)  MODE(RDWR)
```

Which gets resolved to:

```
MOUNT FILESYSTEM('HFS.SC64.DEV') MOUNTPOINT('/SC64/dev') NOAUTOMOVE TYPE(HFS)
MODE(RDWR)
```

This mounts the HFS.**SC64**.DEV file on mount point **SC64**/dev.

The /etc, /var and /tmp files get mounted this way. When the system tries to access any of these files the symlinks in the sysplex root get resolved to /**SC64**/dev, /**SC64**/etc, /**SC64**/var and /**SC64**/tmp and it picks the correct file for the system.

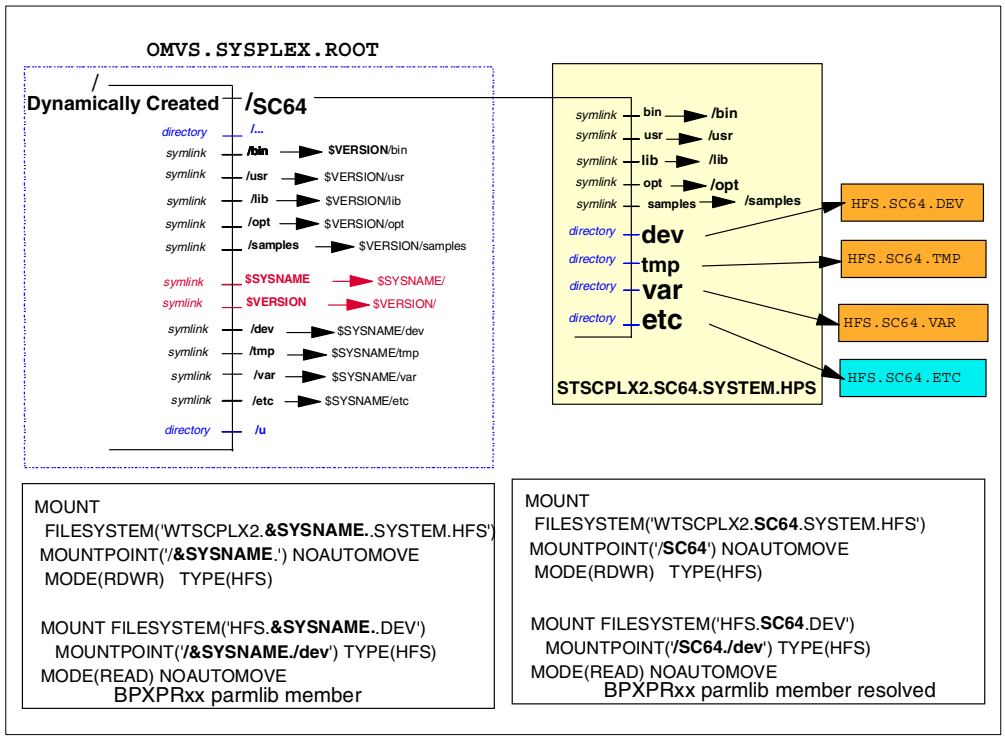


Figure 108. System Specific HFS Set up

10.1.4 OS/390 2.9 system *without* HFS sysplex sharing

Please note that if you want to upgrade your system to OS/390 2.9, but do not want to do sysplex sharing, you do not need to make any changes to your BPXPRMxx member. See Figure 109 for the HFS structure prior to OS/390 2.9.

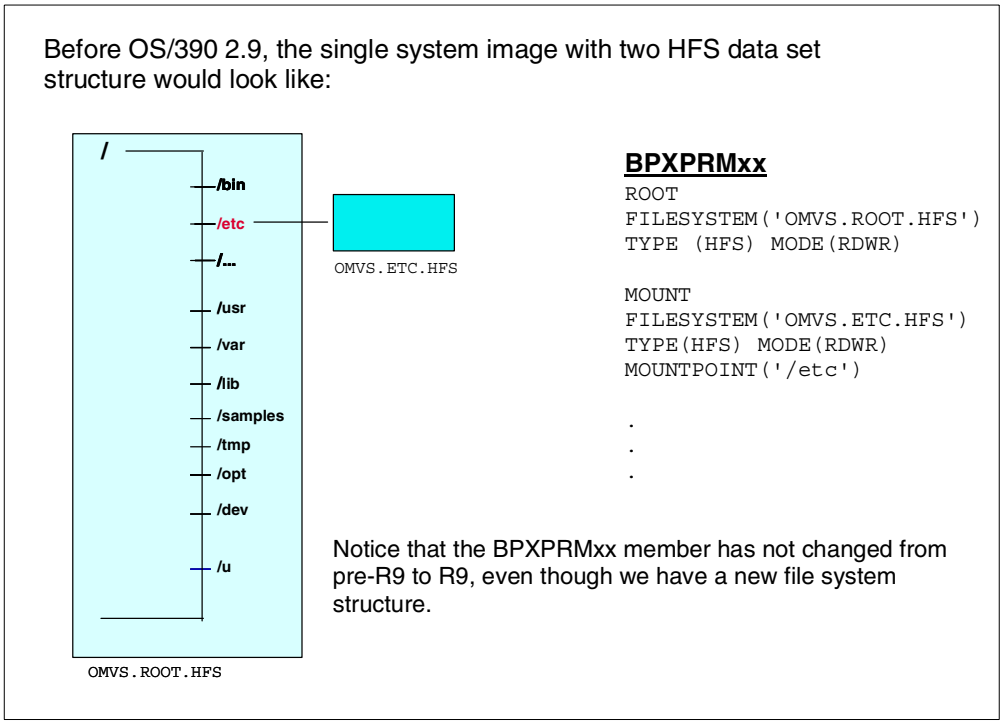


Figure 109. Structure of HFS before OS/390 2.9

10.1.5 Single system image with read-only root

We recommend that you have a read-only root file system. You should have the customizable files /tmp, /dev, /var and /etc mounted off the mount point /SYSTEM. If you follow the recommendation, mounting of these files will occur as before, even with the new sysplex root structure. So, you will not have to make any changes to the BPXPRMxx member. Please see Figure 110 for the structure for a single 2.9 system with a read-only root.

Note

ServerPac and CBPDO customers have to run a job BPXISETS during system installation to convert /etc to a symlink $SYSNSAME/etc.

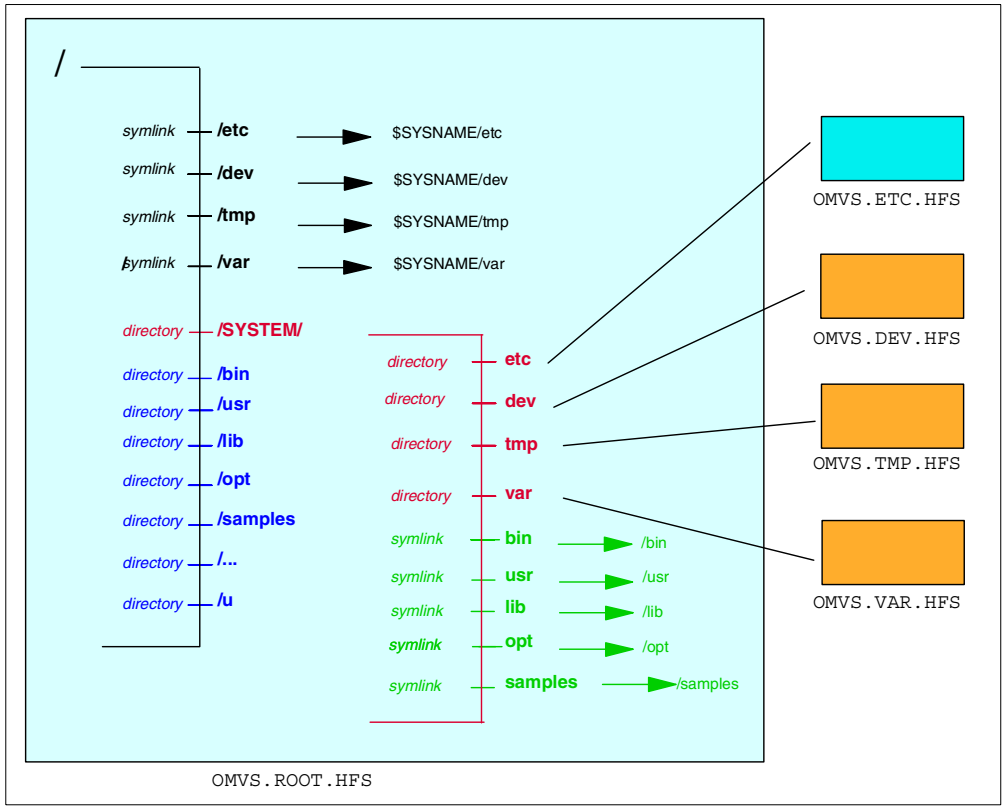


Figure 110. Single OS/390 2.9 system with a read-only root

10.1.6 Multiple systems on different releases

The systems in the sysplex could be running different software levels. Also the systems could have different customization. In such an environment you will have three main sysplex wide HFS data sets. These are the sysplex root, system specific HFS and version specific HFS.

The sysplex root is visible from each system as the main root. As described earlier, it contains symlinks to redirect references to system-specific and version-specific HFS data sets to the correct mount point. The sysplex root is a small file. This should be mounted as read/write because the mount points for system and version HFS are dynamically created on the sysplex root when a system joins the sysplex.

You need one system specific HFS for each system containing the customized /dev,/ tmp, /var and /etc files. You should mount the system specific HFS as read/write.

You will have one version specific HFS data set for each release or service level of OS/390. This is the IBM supplied HFS file containing all the system libraries. You should not modify this HFS data set and should mount it read only.

We describe three scenarios:

1. The first system in the sysplex
2. Two systems in sysplex sharing the same release or version of software
3. Two systems in sysplex with different releases or versions of software

10.1.6.1 The first system in the sysplex

Please see in Figure 111 that the sysplex root directs the references to system specific HFS files /dev,/etc, /var and /tmp to the HFS data set OMVS.SY1.SYSTEM.HFS mounted on mount point SY1. The version specific files /bin, /lib and /samples are directed to OMVS.ROOT.HFS mounted on REL9.

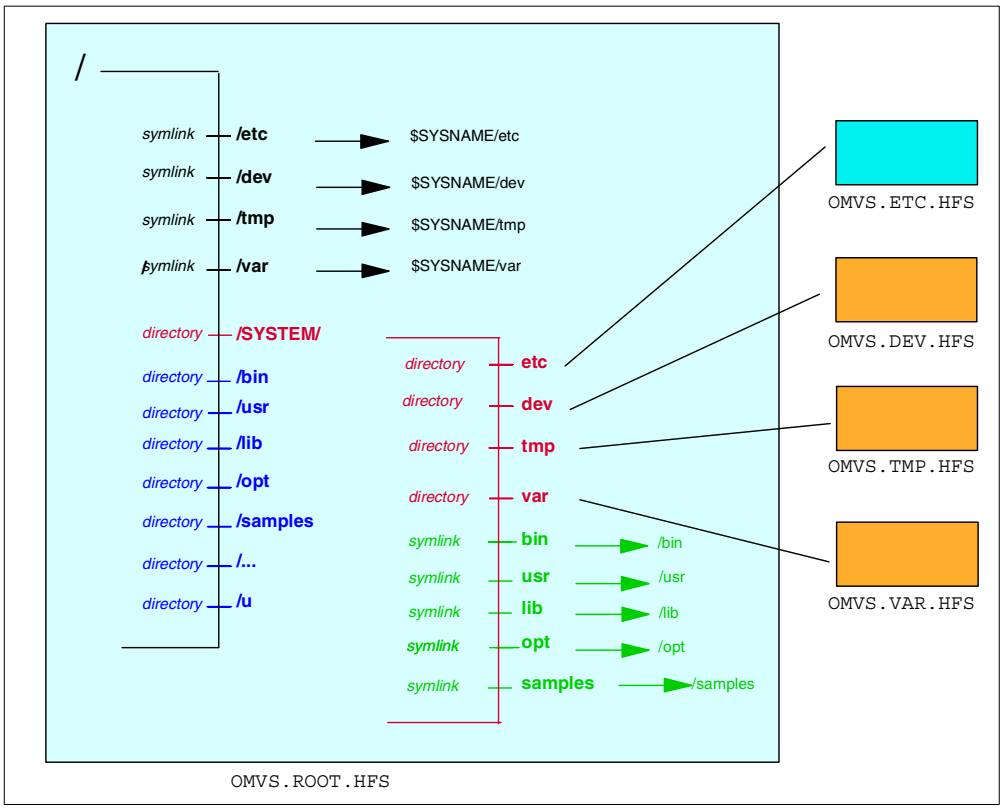


Figure 111. The first system in the sysplex

10.1.6.2 Multiple systems in sysplex sharing the same software version

Please see Figure 112. Systems SY1 and SY2 share the same version HFS in OMVS.ROOT.HFS mounted at REL9. Both these systems would either have VERSION(REL9) in their BPXPRMxx members or they would be sharing the same BPXPRMxx member.

SY1 has its own specific HFS data set OMVS.SY1.SYSTEM.HFS and is mounted on /SY1. This mount point is created dynamically on the sysplex root when SY1 is IPLed. Similarly SY2 has OMVS.SY2.SYSTEM.HFS mounted on /SY2.

When the system SY2 refers to any of its system specific file like /etc, it appears as $SYSNAME/etc to system SY2.This gets resolved to /SY2/etc and so SY2 picks the correct file.

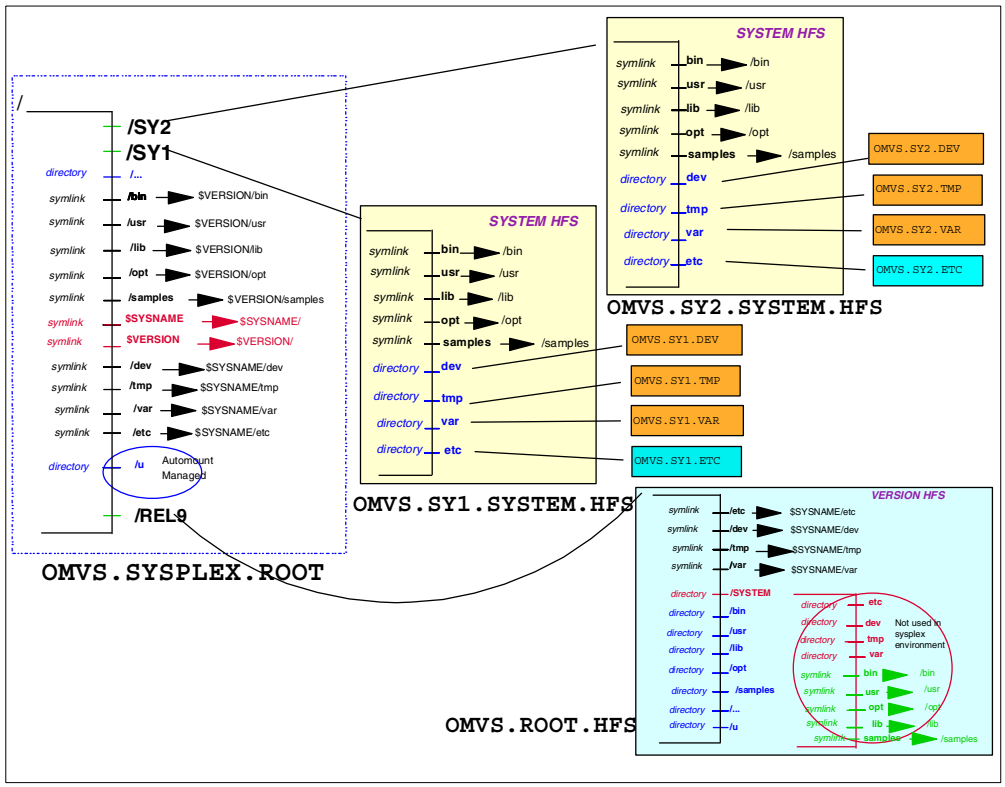


Figure 112. Two systems in sysplex sharing same version HFS

10.1.6.3 Multiple systems with different versions of software

In this case we have two systems SY1 and SY2. But we have two version HFS data sets (REL9 and REL10) available in the sysplex. The systems SY1 and SY2 can use either REL9 or REL10 by specifying it in the version statement of BPXPRMxx parmlib member. Figure 113 illustrates this.

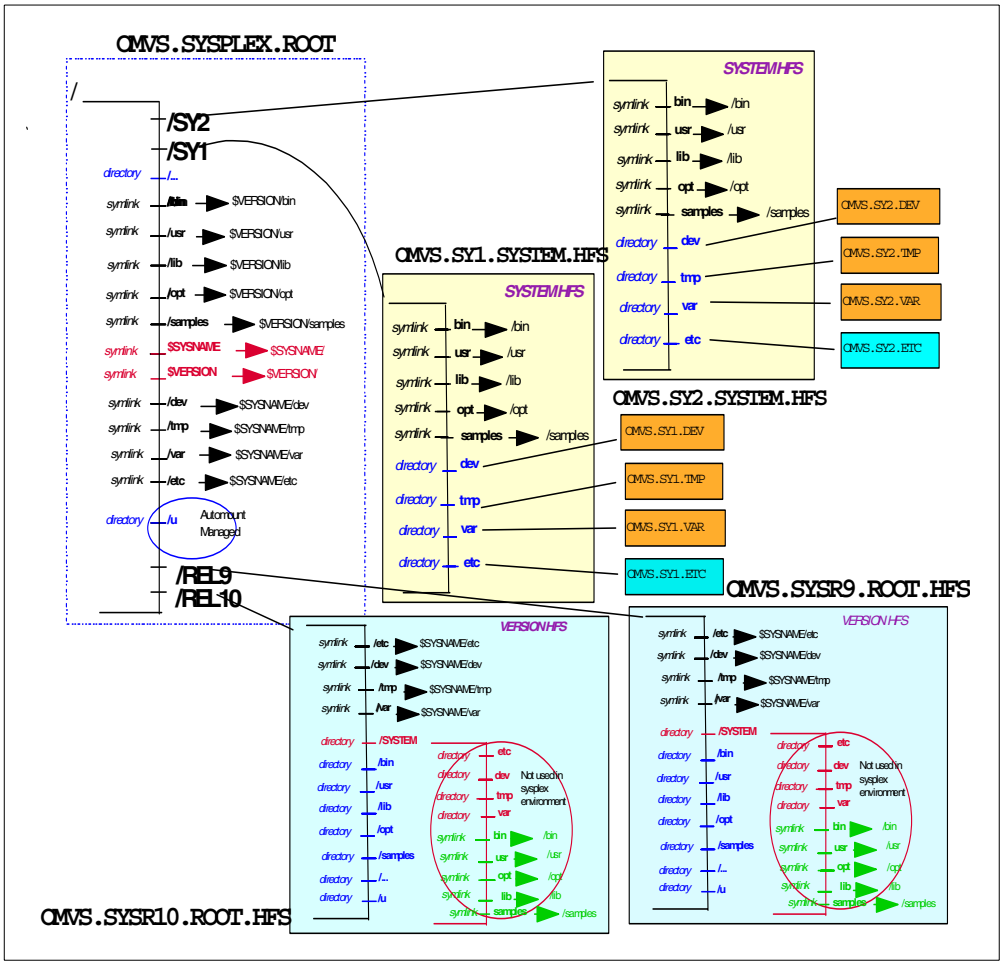


Figure 113. Two systems and two versions

10.1.7 System joins sysplex

Now assume that a third system SY3 joins the sysplex. When that system is IPLed a new mount point SY3 is created in the sysplex root. You specify the system specific HFS in mount statement of the BPXPRMxx to be mounted at this mount point.

In the version statement in BPXPRMxx you will specify which version of HFS to be used. So when the system SY3 is IPLed into the sysplex it will refer to the correct system specific files and version specific files.

10.1.8 System leaves sysplex

When a system leaves the sysplex, normally the system specific files also disappear with the system and will not be available to the other systems in the sysplex. However, it is possible to specify the AUTOMOVE parameter in the mount statement of an HFS data set. This will transfer ownership of the data set to another system before leaving the sysplex so that this data set will still be available to other systems in the sysplex.

Normally data sets are owned by systems on which they are mounted. However, you can specify a different owner to a data set in the mount command.

10.2 Implementation of HFS sysplex sharing

We assume that you have already installed OS/390 2.9 on all the systems that will participate in HFS sharing and the systems are already set up in a sysplex.

For more details, please refer to Chapter 17 *"Shared HFS in a Sysplex"* in *OS/390 V2R9.0 Unix System Services Planning,* SC28-1890*.*

Implementation for sysplex-wide HFS sharing consists of these six steps:

1. Create OMVS Couple data set
2. Define the Couple data set to XCF
3. Create Sysplex Root
4. Create system specific HFS data sets
5. Dynamically add OMVS Couple data sets to XCF
6. IPL the systems

These steps are explained in detail in this chapter. For every step we provide information on:

- What is accomplished by the step
- The options available to you
- The questions you need to ask yourself
- The issues you need to consider
- How we implemented the step
- Listings of the JCL and job output for our implementation

10.2.1 Step 1 - Create OMVS couple data set

In this step, you define the data structure named BPXMCDS. This is used by XCF to maintain the necessary information to allow HFS sharing across the Sysplex.

When you create this data set you need to specify the maximum number of HFS mounts and automount rules you want to support under OMVS. This is the total number sysplex-wide. You should set this value right for your installation. If you set it too high and more than what you really need, the HFS performance may be affected. Please refer to 8.4.3, “How shared HFS affects mount times” on page 205 for a discussion on this topic.

You also specify the name of your sysplex that will be sharing the HFS and the maximum number of systems in the sysplex.

You should define two couple data sets: one as primary and the other as secondary. Define these on two different volumes for availability reasons. These may or may not be cataloged.

Use the sample JCL supplied in SYS1.SAMPLIB(BPXISCDS) and customize it for this step.

We modified the sample JCL as below to define the Couple data sets. The changes we made to the SAMPLIB member are shown in **bold.**

```
/* Begin definition for OMVS couple data set (1)                    */
    DEFINEDS SYSPLEX(SANDBOX)         /* Name of the sysplex in
                                         which the OMVS couple data
                                         set is to be used.         */
             DSN(SYS1.XCF.OMVS00) VOLSER(SBOX03) /* The name and
                                         volume for the OMVS
                                         couple data set. The
                                         utility will allocate a
                                         new data set by the name
                                         specified on the volume
                                         specified.                 */
             MAXSYSTEM(8)             /* Number of systems in the
                                         sysplex to be supported by
                                         this couple data set. Default
                                         value is eight.        @01A*/
             CATALOG                /* Default is not to CATALOG.
                                                                    @01C*/
             DATA TYPE(BPXMCDS)       /* The type of data in the
                                         data set being created is
                                         for OMVS. BPXMCDS is the
                                         TYPE for OMVS.             */
             ITEM NAME(MOUNTS) NUMBER(500) /* Specifies the number of
                                         MOUNTS that can be supported
                                         by OMVS.
                                         Default =   100
                                         Minimum =     1
                                         Maximum = 50000        @D1C*/
             ITEM NAME(AMTRULES) NUMBER(50) /* Specifies the number
                                         of automount rules that can
                                         be supported by OMVS.
                                         Default =    50
                                         Minimum =    50
                                         Maximum =  1000        @D1A*/

/* Begin definition for OMVS couple data set (2)                    */
   DEFINEDS SYSPLEX(SANDBOX)          /* Name of the sysplex in
                                         which the OMVS couple data
                                         set is to be used.         */
            DSN(SYS1.XCF.OMVS01) VOLSER(SBOX04) /* The name and
                                         volume for the OMVS
                                          couple data set. The
                                          utility will allocate a
                                          new data set by the name
                                          specified on the volume
                                          specified.                */
             MAXSYSTEM(8)             /* Number of systems in the
                                         sysplex to be supported by
                                         this couple data set. Default
                                         value is eight.        @01A*/
             CATALOG                  /* Default is not to CATALOG.
                                                                    @01C*/
             DATA TYPE(BPXMCDS)       /* The type of data in the
                                         data set being created is
                                         for OMVS. BPXMCDS is the
                                         TYPE for OMVS.             */
              ITEM NAME(MOUNTS) NUMBER(500) /* Specifies the number of
                                         MOUNTS that can be supported
                                         by OMVS.
                                         Default =   100
                                         Minimum =    10
                                         Maximum = 50000        @D1C*/
              ITEM NAME(AMTRULES) NUMBER(50) /* Specifies the number
```

The output of the job follows. Again we show here only the relevant portion of the output.

```
IGD100I 253A ALLOCATED TO DDNAME COUPLE   DATACLAS (        )
IXC292I DATA SET FORMATTING COMPLETE: DATA SET REQUIRES 32 TRACKS ON VOLSER SBOX03

IXC292I 1 RECORDS FORMATTED WITH 500 MOUNTS ITEMS EACH
IXC292I 1 RECORDS FORMATTED WITH 50 AMTRULES ITEMS EACH
IGD100I 253B ALLOCATED TO DDNAME COUPLE   DATACLAS (        )
IXC292I DATA SET FORMATTING COMPLETE: DATA SET REQUIRES 32 TRACKS ON VOLSER SBOX04

IXC292I 1 RECORDS FORMATTED WITH 500 MOUNTS ITEMS EACH
IXC292I 1 RECORDS FORMATTED WITH 50 AMTRULES ITEMS EACH
```

For more details on how to define Sysplex Couple data sets, please refer to section 3.1 on “Considerations for All Couple Data Sets", in *OS/390 V2R9.0 MVS setting Up a Sysplex,* GC28-1779.

10.2.2 Step 2 - Define the OMVS couple data sets to XCF

Next you need to tell XCF the names of the primary and secondary couple data sets you have just defined. This is done by updating the COUPLExx member in your PARMLIB data set. The COUPLExx member contains information on all couple data sets used by your sysplex. You need to add the OMVS Couple data set information to the COUPLExx member. We added the following statements to our COUPLExx member shown by **bold** letters.

```
COUPLE SYSPLEX(SANDBOX)
       PCOUPLE(SYS1.XCF.CDS00)
       ACOUPLE(SYS1.XCF.CDS01)
       CLEANUP(30)
       RETRY(10)
/* DEFINITIONS FOR CFRM POLICY */
DATA TYPE(CFRM)
     PCOUPLE(SYS1.XCF.CFRM00)
     ACOUPLE(SYS1.XCF.CFRM01)
/* DATASETS FOR SFM POLICY */
DATA TYPE(SFM)
     PCOUPLE(SYS1.XCF.SFM00)
     ACOUPLE(SYS1.XCF.SFM01)
/* DATASETS FOR WLM POLICY */
DATA TYPE(WLM)
     PCOUPLE(SYS1.XCF.WLM00)
     ACOUPLE(SYS1.XCF.WLM01)
DATA TYPE(BPXMCDS)
      PCOUPLE(SYS1.XCF.OMVS01)
      ACOUPLE(SYS1.XCF.OMVS02)
/* LOCAL XCF MESSAGE TRAFFIC */
LOCALMSG MAXMSG(512) CLASS(DEFAULT)
/* PATH DEFINITIONS FOR DEFAULT SIGNALLING */
```

10.2.3 Step 3 - Create sysplex root HFS

You can customize the JCL in SYS1.SAMPLIB(BXISYSR) and run it to create the sysplex root. This job allocates the data set and runs a REXX exec BPXISYS1 to create the necessary directories and symlinks in the data set.

You need superuser authority to run this job.

We customized the BXISYSR exec as follows. The changes are shown in **bold.**

This data set may or may not be SMS managed. **The sysplex root must be mounted in read/write mode.**

Important Information

We recommend that the sysplex root be kept very stable; Updates and changes to the sysplex root should be made as infrequent as possible.

```
//IKJEFT1A  EXEC PGM=IKJEFT1A,PARM='BPXISYS1'
//*
//ROOTSYSP  DD DSNAME=WTSCPLX2.SYSPLEX.ROOT,
//          DISP=(,CATLG),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(1,0,1)),
//*         STORCLAS=OPENMVS
//          UNIT=3390,VOL=SER=SBOX02
//*
//SYSEXEC   DD DSN=SYS1.SAMPLIB,DISP=SHR ,
//*         UNIT=SYSALLDA,VOL=SER=tvol2
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//*
```

```
The EXEC BPXISYS1 ran at  17:12:17  on  13 Mar 2000 .

This EXEC completed with Return Code 0 .

Created the following directories:
==================================
...
u

Problems creating the following directories:
============================================
No problems while creating the directories.

Created the following symlinks:
===============================
bin   --->   $VERSION/bin
usr   --->   $VERSION/usr
lib   --->   $VERSION/lib
opt   --->   $VERSION/opt
samples   --->   $VERSION/samples
$VERSION   --->   $VERSION/
$SYSNAME   --->   $SYSNAME/
dev   --->   $SYSNAME/dev
tmp   --->   $SYSNAME/tmp
var   --->   $SYSNAME/var
etc   --->   $SYSNAME/etc

Problems creating the following symlinks:
=========================================
No problems while creating the symlinks.

End of EXEC.
```

The output of the job shows the directories and the symlinks created. Please compare this output with Figure 105 on page 222 to see what the structure of the sysplex root will look like after it is created.

10.2.4 Step 4 - Create system specific HFS

You will need to define system specific HFS files for each system in the sysplex. We recommend that you use a naming convention that will associate this data set to the system for which it is defined. The sysplex root has symbolic links to mount points for system-specific etc, var, tmp, and dev HFS data sets.

Important Information

The system-specific HFS data set should be mounted read/write. In addition, we recommend that the name of the system-specific data set contain the system name as one of the qualifiers. This allows you to use the &SYSNAME symbolic in BPXPRMxx.

The mount point for these data sets will be dynamically created by the system during OMVS startup.

You can customize the JCL in SYS1.SAMPLIB(BPXISYSS) to create system specific HFS. You need to run this job for each system in the sysplex that will be sharing HFS. This dataset may be SMS managed or non-SMS managed.

We customized the JCL in SAMPLIB as follows. The changes are in **bold**.

We defined the data set for two systems SC64 and SC65. Only SC64 JCL is shown

We defined the SC64 HFS data set as SMS managed and the SC65 HFS as non-SMS managed

The output of the job is also shown below. It runs a REXX exec BPXISYS2 to define the directories and sysmlinks,

.

```
//IKJEFT1A  EXEC PGM=IKJEFT1A,PARM='BPXISYS2'
//*
//HFSSYSTS  DD DSNAME=WTSCPLX2.SC64.SYSTEM.HFS,
//          DISP=(,CATLG),
//          DSNTYPE=HFS,
//          SPACE=(CYL,(1,0,1)),
//          STORCLAS=OPENMVS
//*         UNIT=uuuu,VOL=SER=vvvvvv
//*
//SYSEXEC   DD DSN=SYS1.SAMPLIB,DISP=SHR ,
//*         UNIT=SYSALLDA,VOL=SER=tvol2
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//*
```

```
The EXEC BPXISYS2 ran at  17:16:17  on  13 Mar 2000 .

This EXEC completed with Return Code 0 .

Created the following directories:
==================================
dev
tmp
var
etc

Problems creating the following directories:
============================================
No problems while creating the directories.

Created the following symlinks:
===============================
bin   --->   /bin
usr   --->   /usr
lib   --->   /lib
opt   --->   /opt
samples   --->   /samples

Problems creating the following symlinks:
=========================================
No problems while creating the symlinks.



End of EXEC.
```

10.2.5 Step 5 - Update BPXPRMxx parmlib member

You specify the various parameters to control the file system in the BPXPRMxx parmlib member. This member also contains information to control the OMVS set up and processing. We recommend that you use two different members: one containing the file system information and the other containing OMVS information. Then you can update the OMVS parameter of IEASYSxx parmlib member to point to both the BPXPRMxx members. You will find that migrating from one release to another is easier if you use two members for BPXPRMxx.

You can use a common BPXPRMxx member for all the systems in the sysplex.

There are four parameters in BPXPRMxx that are mainly relevant to HFS sharing in a sysplex. These are:

1. SYSPLEX
2. VERSION
3. ROOT
4. MOUNT

10.2.5.1 SYSPLEX

You should specify SYSPLEX(YES) to share HFS across the Sysplex. It is this parameter which tells the system at the time of IPL to take part in HFS sharing across the sysplex.

10.2.5.2 VERSION

This statement will dynamically create a mount point at the time of IPL to mount the Version HFS file. The version HFS is the IBM-supplied root HFS data set. You should specify a version parameter which identifies your OS/390 system release level. The system residence volser might be an appropriate name you can use. Different MVS systems in the sysplex can specify different VERSION parameters in their BPXPRMxx member to allow different releases or service levels of root HFS. IBM recommends that you mount the Version HFS in read-only mode. For specific actions you have to take before you can mount the Version HFS in read-only mode please see the section on "Post Installation Actions for Mounting the Root HFS in Read-Only" in *OS/390 V2R9.0 UNIX System Services Planning,* SC28-1890.

Important Information

We do not recommend using &SYSNAME as one of the qualifiers for the version HFS data set name. Appropriate names may be the name of the target zone, &SYSR1, or any other qualifier meaningful to the system programmer.

10.2.5.3 ROOT

Here you specify the name of the Sysplex Root data set you created in step 3. This data set must be mounted read/write.

10.2.5.4 MOUNT

You should specify the mount information for all the HFS files you will require for your system. If you used &SYSNAME as one of the qualifiers when you defined your system specific HFS data sets in step 4, then you can create a single BPXPRMxx member for all systems in your sysplex.

You can specify two new parameters for the MOUNT statements.The SYSNAME parameter specifies the name of the MVS system in the Sysplex that should own the HFS data set being mounted. The AUTOMOVE parameter determines if another system in the sysplex can take ownership of the HFS data set if the owning system goes down.

We used a common BPXPRMxx member for both the systems. Below is a copy of the BPXPRMxx member we used. It shows only the relevant part of the BPXPRMxx member.

Please note that you have to specify &SYSNAME for the mount point for TMP, so that each system gets its own mount point for the TMP file system.

```
FILESYSTYPE TYPE(HFS)            /* Type of file system to start  */
             ENTRYPOINT(GFUAINIT) /* Entry Point of load module    */
             PARM(' ')            /* Null PARM for physical file
                                     system                        */
          /* ASNAME(adrspc01) */  /* Name of address space for
                                     physical file system          */
VERSION('&SYSR1.')
 SYSPLEX(YES)

 ROOT     FILESYSTEM('WTSCPLX2.SYSPLEX.ROOT')
          TYPE(HFS)               /* TYPE OF FILE SYSTEM           */
          MODE(RDWR)              /* (OPTIONAL) CAN BE READ OR RDWR.
                                        DEFAULT = RDWR               */

 MOUNT FILESYSTEM('WTSCPLX2.&SYSNAME..SYSTEM.HFS')
       MOUNTPOINT('/&SYSNAME.')
       NOAUTOMOVE
       TYPE(HFS)  MODE(RDWR)

 MOUNT FILESYSTEM('HFS.OS390R9.&SYSR1..ROOT')
       MOUNTPOINT('/$VERSION')
       TYPE(HFS)  MODE(READ)

 MOUNT FILESYSTEM('HFS.&SYSNAME..DEV')
       MOUNTPOINT('/&SYSNAME./dev')
       NOAUTOMOVE
       TYPE(HFS)  MODE(RDWR)

MOUNT FILESYSTEM('HFS.&SYSNAME..ETC')
      MOUNTPOINT('/&SYSNAME./etc')
      NOAUTOMOVE
      TYPE(HFS)  MODE(RDWR)

MOUNT FILESYSTEM('HFS.USERS')
      MOUNTPOINT('/u')
      NOAUTOMOVE
      TYPE(HFS)  MODE(RDWR)

MOUNT FILESYSTEM('HFS.&SYSNAME..VAR')
      MOUNTPOINT('/&SYSNAME./var')
      NOAUTOMOVE
      TYPE(HFS)  MODE(RDWR)

FILESYSTYPE TYPE(TFS) ENTRYPOINT(BPXTFS)
MOUNT FILESYSTEM('/&SYSNAME./TMP')
      TYPE(TFS) MODE(RDWR)
      MOUNTPOINT('/&SYSNAME./tmp')
      PARM('-s 500')
      NOAUTOMOVE

FILESYSTYPE TYPE(AUTOMNT) ENTRYPOINT(BPXTAMD)
```

10.2.6 Step 6 - Dynamically add the OMVS couple data sets to XCF

This step is just to verify that your couple data sets are correct before you IPL the systems in the next step. Issue the system commands shown in order to add the

primary Couple data set, the secondary Couple data set, and then to verify that they have been added successfully:

```
SETXCF COUPLE,TYPE=BPXMCDS,PCOUPLE=XCF,OMVS00

SETXCF COUPLE,TYPE=BPXMCDS,ACOUPLE=XCF,OMVS01

D XCF,COUPLE,TYPE=BPXMCDS
```

The output from these commands is shown below.

```
SETXCF COUPLE,TYPE=BPXMCDS,PCOUPLE=SYS1.XCF.OMVS00
IXC309I SETXCF COUPLE,PCOUPLE REQUEST FOR BPXMCDS WAS ACCEPTED
IEF196I IEF237I 253A ALLOCATED TO SYS00077
IXC286I COUPLE DATA SET 441
SYS1.XCF.OMVS00,
VOLSER SBOX03, HAS BEEN ADDED AS THE PRIMARY
FOR BPXMCDS ON SYSTEM SC64
IEF196I IEF237I 253A ALLOCATED TO SYS00077
IXC286I COUPLE DATA SET 837
SYS1.XCF.OMVS00,
VOLSER SBOX03, HAS BEEN ADDED AS THE PRIMARY
FOR BPXMCDS ON SYSTEM SC63
IEF196I IEF237I 253A ALLOCATED TO SYS00080
SETXCF COUPLE,TYPE=BPXMCDS,ACOUPLE=SYS1.XCF.OMVS01
IXC309I SETXCF COUPLE,ACOUPLE REQUEST FOR BPXMCDS WAS ACCEPTED
IXC260I ALTERNATE COUPLE DATA SET REQUEST FROM SYSTEM 448
SC64 FOR BPXMCDS IS NOW BEING PROCESSED.
DATA SET:  SYS1.XCF.OMVS01
IEF196I IEF237I 253B ALLOCATED TO SYS00078
IEF196I IEF237I 253B ALLOCATED TO SYS00078
IXC286I COUPLE DATA SET 521
SYS1.XCF.OMVS00,
VOLSER SBOX03, HAS BEEN ADDED AS THE PRIMARY
FOR BPXMCDS ON SYSTEM SC65
IXC251I NEW ALTERNATE DATA SET 450
SYS1.XCF.OMVS01
FOR BPXMCDS HAS BEEN MADE AVAILABLE
D XCF,COUPLE,TYPE=BPXMCDS
 IXC358I  14.06.26  DISPLAY XCF 452
 BPXMCDS COUPLE DATA SETS
 PRIMARY    DSN: SYS1.XCF.OMVS00
            VOLSER: SBOX03     DEVN: 253A
            FORMAT TOD         MAXSYSTEM
            03/09/2000 18:01:40        8
            ADDITIONAL INFORMATION:
             NOT PROVIDED
 ALTERNATE  DSN: SYS1.XCF.OMVS01
            VOLSER: SBOX04     DEVN: 253B
            FORMAT TOD         MAXSYSTEM
            03/09/2000 18:01:41        8
            ADDITIONAL INFORMATION:
             NOT PROVIDED
 BPXMCDS IN USE BY ALL SYSTEMS
```

10.2.7 IPL the systems

This is to activate the COUPLExx and BPXPRMxx changes. Now you are ready to share HFS data sets across the sysplex.

10.2.8 Implementation summary

The following pictures summarize the installation. Figure 114 shows the relationship between XCF, the couple data sets and COUPLExx parmlib member.

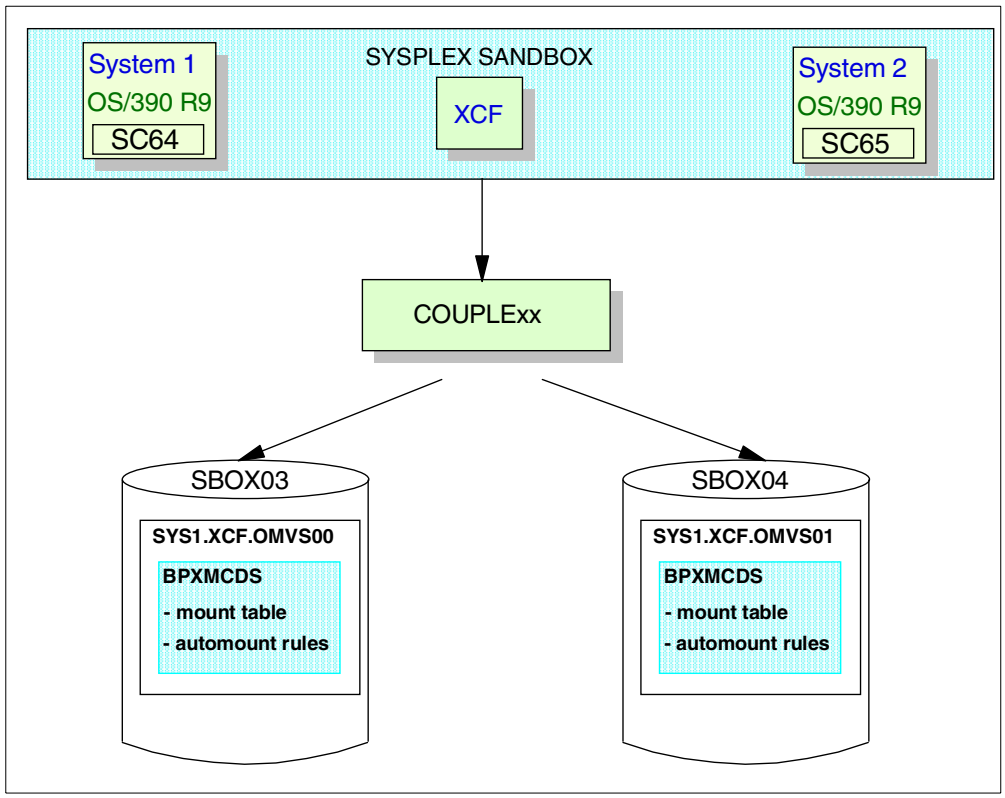


Figure 114. XCF and the Couple data sets

Figure 115 shows the relationship between the sysplex root, the version HFS file, the system specific HFS and the BPXPRMxx parmlib member.

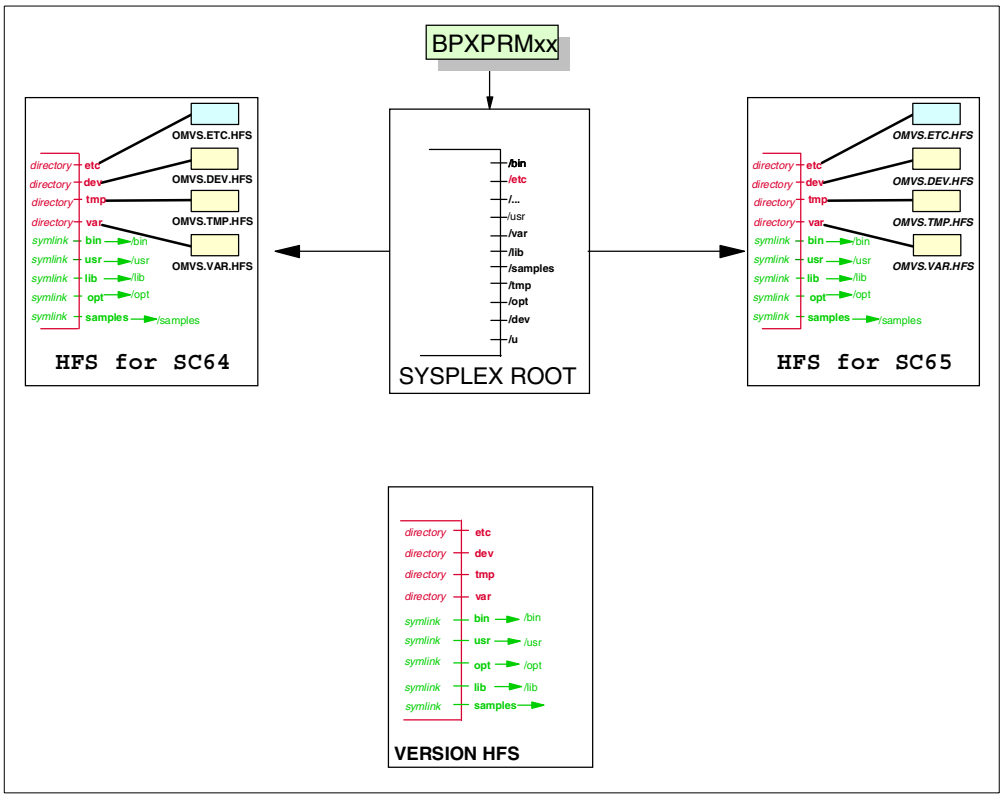


Figure 115. BPXPRMxx and HFS systems

10.3 How to add another system to the sysplex for HFS sharing

To add another system to share HFS data in your sysplex, you need to:

1. Make sure that the system you are adding is already running OS/390 2.9.
2. Update the COUPLExx parmlib member to include the OMVS CDS you have defined for HFS sharing. You can also use a common COUPLExx member for all the systems in the sysplex. Please see 10.2.2, “Step 2 - Define the OMVS couple data sets to XCF” on page 234.
3. Define system specific HFS data sets for the system as described in 10.2.4, “Step 4 - Create system specific HFS” on page 236.
4. Dynamically add the OMVS CDS as in 10.2.6, “Step 6 - Dynamically add the OMVS couple data sets to XCF” on page 240.
5. IPL the system into the sysplex.

10.3.1 Change in ls command

The *ls* command will now require you to specify "ls /dev/". Please notice the trailing slash. Prior to OS/390 2.9, you could say "ls /dev" and you would get a listing of the /dev directory. To minimize this difference, if you wish, you could place alias ls=ls -L in your /etc/profile file.

10.3.2 Automount facility

The automount facility gives you the flexibility to mount files as and when they are required rather than keeping them mounted all the time. The mount point directories are internally created as they are required. Later, when the file system is no longer in use the mount point directories are deleted. You specify the details on how to and when to mount and unmount the file in a policy file.

In the sysplex environment, there is a common mount table for all systems in the sysplex. This resides in the OMVS CDS.

Important information

We strongly recommend that you keep automount policies consistent across all the systems in the sysplex. At any time, the most recent automount policy that was loaded prevails for all the systems participating in the sysplex.

Automount delay is the time for which you want to keep an auto-mounted file mounted after its last use. This is specified in your MapName File that contains the mount information for your automount files. The default automount delay is 0 minutes. In a sysplex environment we do not recommend that you use the default value. You should specify a delay time of at least 10 minutes.

Automount does not support the new key words AUTOMOVE and SYSNAME.

10.3.3 Special files

You cannot share character-special files nor UNIX domain socket address files in read/write mode among systems participating in the sysplex.

You can share FIFO special files between systems.

10.3.4 Cloning systems

OS/390 2.9 now supports non-SMS HFS data sets. This means that it is now possible for you to allocate HFS data sets on your system residence volume and catalog them with VOLSER=******.

Appendix A. Miscellaneous implementation topics

This chapter discusses miscellaneous implementation topics that storage administrators should know about. The topics covered are:

- The pax, tar, and cpio commands
- The automount facility

A.1 The pax, tar and cpio commands

In MVS, you can use Data Facility System-Managed Storage Hierarchical Storage Manager (DFSMShsm) to back up or archive hierarchical file systems automatically.

In the shell, files are often bundled together in single files by utilities like pax, tar, and cpio. When these files are bundled into a single file, it is called an archive file. Usually the file name of the archive indicates the utility that was used when the file was built. For example, a file named mvsport.tar indicates that the tar utility was used. The three utilities basically provide the same function: reading and writing of archive files.

The important thing to know is that the tar and cpio commands can only read and write files of their respective formats, but pax can read or write in either format. So given a pax, tar, or cpio file, you can use pax from the OS/390 UNIX System Services shell to explode or unwind the archive into its individual files.

To save disk space and transmission time, archive files can also be compressed with the tar, cpio and pax utilities by using the -z option. The naming convention that is generally used for a compressed archive file is to end the file name with a .Z, for example, mvsport.tar.Z.

You can copy archive files to an MVS data set, and then to tape. You can retrieve archive files from a tape into an MVS data set, and then copy them into the file system.

This topic provides information about how to use pax, tar and cpio to back up and restore files into and from MVS data sets.

A.1.1 Using pax to back up and restore files

The pax command can read and write files in:

- CPIO format
- ASCII format
- CPIO binary format
- TAR format
- USTAR format

It can read files that were written using tar, cpio, or pax itself. How it handles file name length and preservation of link information across the backup and restore process depends on the format you select: if you select CPIO, it behaves like the cpio command; and if you select TAR, it behaves like the tar command.

For information about preserving links with pax, and about the CPIO and TAR archive file formats, see the *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892.

In the following examples, /tmp/posix is the working directory.

A.1.1.1 Backing up a complete directory into an MVS data set

To back up a complete directory into a data set, including the subdirectories and their contents, enter these two commands:

```
pax -wf archive_file directory_name
tso "OGET 'archive_file' 'DATA_SET_NAME' BINARY"
```

Here, directory_name is the name of the directory you want to archive, 'archive_file' is an absolute pathname and 'DATA_SET_NAME' is a fully qualified data set name. The `pax` command creates an archive file with the specified name in the current working directory. The OGET command copies the archive file into the specified MVS data set.

For example, these two commands back up the directory /tmp/posix/testpgm into the data set POSIX.TESTPGM.PAX:

```
pax -wf testpgm.pax /tmp/posix/testpgm
tso "OGET '/tmp/posix/testpgm.pax' 'POSIX.TESTPGM.PAX' BINARY"
```

For the pax command:

- The -w option writes to the archive file.
- The -f option lets you specify the name of the archive file.

After you have copied the archive file into an MVS data set, you can delete it.

Note: To avoid accidentally including the pax file when you create it (which will put you in a loop), you must do *one* of the following:

- Explicitly specify the directories you want included (as in the above example).
- Write the file to a different directory than the one you are archiving. This example archives the contents of the current directory and writes the archive in another directory, /tmp:

  ```
  pax -w . > /tmp/pax.file
  ```

A.1.1.2 Restoring a complete directory from an MVS data set

To restore the directory in the previous example, enter two commands:

```
tso "OPUT 'POSIX.TESTPGM.PAX' '/tmp/posix/testpgm.pax' BINARY"
pax -rf testpgm.pax
```

The OPUT command copies the data set containing an archive file into the specified directory in the file system. The pax command restores the contents of the archive so that they can be accessed in the file system.

For the pax command, the -r option reads from the file specified with the -f option.

A.1.2 Using tar to back up and restore files

The tar command reads and writes headers in either the original TAR format from UNIX systems or the USTAR format defined by the POSIX 1003.1 standard. With the TAR format, the length of the path name you can specify is 100 characters.

With the USTAR format, the length of the path name you can specify is 255 characters. For information on the TAR archive file formats, see the *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892.

During the backup or restore process, tar preserves link information.

If you will be putting the archive file on a tape, the block size that was used when writing the file should be used when reading the file.

In the following examples, /tmp/posix is the working directory.

A.1.2.1 Backing up a complete directory into an MVS data set

To back up a complete directory, including the subdirectories and their contents, into a data set, enter these two commands:

```
tar -cf archive_file directory_name
tso "OGET 'archive_file' 'DATA_SET_NAME' BINARY"
```

where 'archive_file' is an absolute path name and 'DATA_SET_NAME' is a fully qualified data set name.

The tar command creates the specified archive file in your working directory. The OGET command copies the archive file into the specified MVS data set.

For example, the following commands back up the directory /tmp/posix/testpgm into the data set POSIX.TESTPGM.TAR:

```
tar -cvf testpgm.tar /tmp/posix/testpgm
tso "OGET '/tmp/posix/testpgm.tar' 'POSIX.TESTPGM.TAR' BINARY"
```

For the tar command:

- The -c option creates an archive.
- The -v option displays each file name as it processes the archive.
- The -f option uses a specified file name for the archive file.

After you have copied the archive file into an MVS data set, you can delete it.

A.1.2.2 Restoring a complete directory from an MVS data set

To restore the directory in the previous example, enter these two commands:

```
tso "OPUT 'POSIX.TESTPGM.TAR' '/tmp/posix/testpgm.tar' BINARY"
tar -xvf testpgm.tar
```

The OPUT command copies the data set containing an archive file into the specified directory in the file system. The tar command restores the contents of the archive so that they can be accessed in the file system. For the tar command, the -x option restores files from the archive.

A.1.3 Using cpio to back up and restore files

cpio reads and writes either a compact binary format header or an ASCII format header. The cpio command has no limit on the length of a file name. For information on the cpio archive file format, see *OS/390 UNIX System Services Command Reference,* SC28-1892.

In these examples, /tmp/posix is the working directory.

A.1.3.1 Backing up a complete directory into an MVS data set

Backing up a complete directory, including the subdirectories and their contents, into a data set takes two commands:

```
find directory_name -print | cpio -o > archive_file
tso "OGET 'archive_file' 'DATA_SET_NAME' BINARY"
```

where 'archive_file' is an absolute path name and 'DATA_SET_NAME' is a fully qualified data set name.

The find command extracts path names from the specified directory. Its contents are piped to cpio, which creates an archive file in your working directory. The OGET command copies the archive file into the specified MVS data set. We recommend using BINARY for an archive file. For example, to back up the directory /tmp/posix/testpgm into the data set POSIX.TESTPGM.CPIO, enter these two commands:

```
find /tmp/posix/testpgm -print | cpio -o > testpgm.cpio
tso "OGET '/tmp/posix/testpgm.cpio' 'POSIX.TESTPGM.CPIO' BINARY"
```

For the cpio command, the -o option writes to an archive file — in this case, to testpgm.cpio.

After you have copied the archive file into an MVS data set, you can delete it.

A.1.3.2 Restoring a complete directory from an MVS data set

The following commands would restore the directory in the previous example:

```
tso "OPUT 'POSIX.TESTPGM.CPIO' '/tmp/posix/testpgm.cpio' BINARY"
cpio -iud < testpgm.cpio
```

The OPUT command copies the data set containing an archive file into the specified directory in the file system. The cpio command restores the contents of the archive so that they can be accessed in the file system.

For the cpio command:

- The -i option reads an archive — in this case, from testpgm.cpio.
- The -u option overwrites any existing file or directory.
- The -d option creates any necessary intermediate directories.

A.2 Automount facility

This topic discusses the customization of the *automount facility* to control all user file systems so that they will be automatically mounted when they are needed.

A.2.1 Customizing the automount facility

The automount facility lets you designate directories as containing only mount points. This is the preferred method of managing user HFS data sets. As each of these mount points is accessed, an appropriate file system is mounted. The mount point directories are internally created as they are required. Later, when the file system is no longer in use, the mount point directories are deleted.

Think of automount as an administrator that has total control over a directory. When a name is accessed in this directory, it checks its policy to see what file system is supposed to be associated with that name. If it finds one, it (logically)

does an mkdir followed by a mount and quietly moves out of the way. Once out of the way, the root directory of that newly mounted file system is now accessed as that name. For example, we create the user1 directory with the mkdir command.

With automount active and the correct automount policy in place, there is no need to create a user1 directory with the mkdir command; the user1 directory will be dynamically allocated and the OMVS.SC64.USER1 data set will be automatically mounted at the /u/user1 mount point. Later, if the /u/user1 file system has not been accessed based on certain criteria in your automount policy, the OMVS.SC64.USER1 data set will automatically be unmounted.

Another advantage of using the automount daemon to manage user file systems (as opposed to adding them to BPXPRMxx) is when a user goes on vacation for two weeks. Their HFS might not be mounted, and so can become a candidate for HSM to migrate the HFS. If the user HFS is defined in the BPXPRMxx PARMLIB member, the HFS will always be mounted, and so the last access date will always be updated. The automount facility is shown in Figure 116.

See *OS/390 V2R9.0 UNIX System Services Planning*, SC28-1890 and *OS/390 V2R9.0 UNIX System Services Command Reference*, SC28-1892 for additional information about this facility.

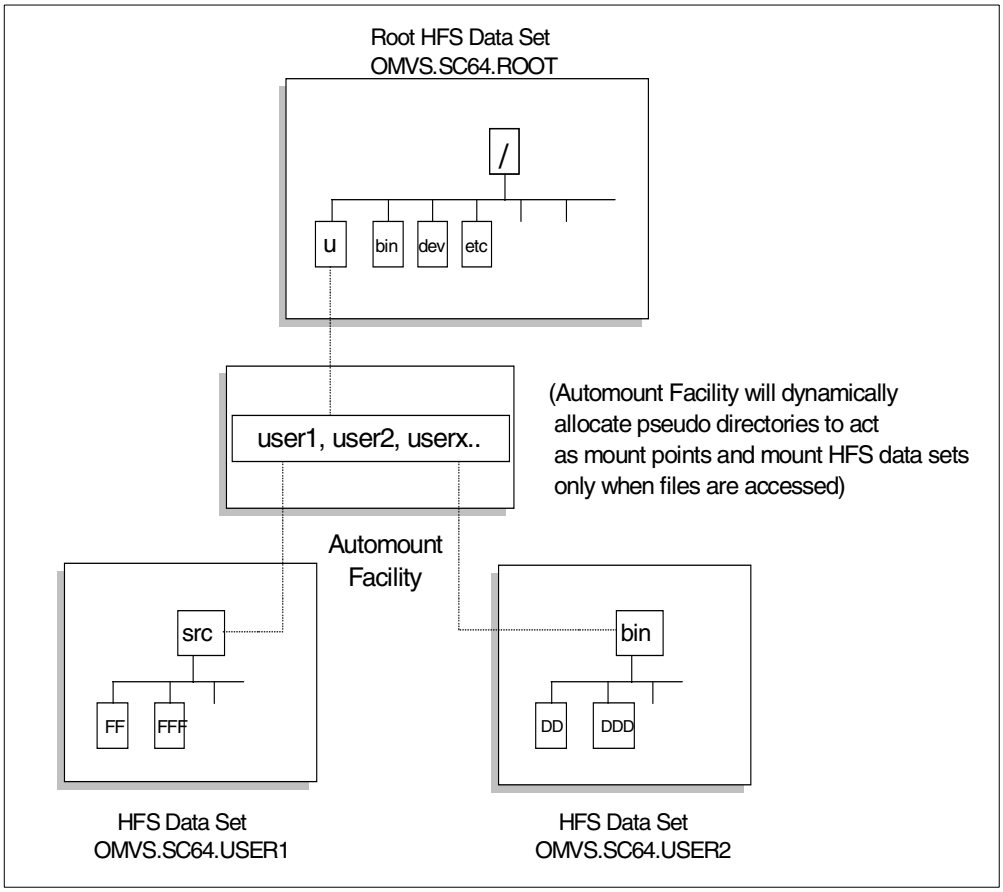


Figure 116. Automount facility

The following steps show how to set up the automount facility to mount user file systems.

A.2.1.1 Step 1 - Modify BPXPRMxx

To use the automount facility, add the following statement to your BPXPRMxx PARMLIB member and re-IPL the MVS system.

```
FILESYSTYPE TYPE(AUTOMNT) ENTRYPOINT(BPXTAMD)
```

A.2.1.2 Step 2 - Customize the definition files

The automount facility uses two definition files, a master file and a MapName file. The default file name of the master file is /etc/auto.master. The /etc/auto.master file contains the directory or directories that will be monitored by automount and the associated MapName files which contain the mount parameters. Figure 117 shows an example of a /etc/auto.master file.

```
BROWSE -- /etc/auto.master ------
COMMAND ===>
*************** Top of Data **********
/u                  /etc/u.map
*************** Bottom of Data *******
```

Figure 117. Example of /etc/auto.master file

The master file defines that automount should manage the /u directory. This means that as soon as someone using UNIX System Services tries to access a directory that is mounted off the /u directory, automount will automatically mount the HFS data set based on the MapName policy in Figure 118.

Note: In order to make the most efficient use of an automount MapName policy that contains generic entries, it is important to come up with a consistent HFS data set naming convention. In our examples here, all our HFS data sets have a high-level qualifier of OMVS and the low-level qualifier is equal to the user name.

```
BROWSE -- /etc/u.map -------------
COMMAND ===>
*************** Top of Data **********
name            *
type            HFS
filesystem      OMVS.SC64.<uc_name>
mode            rdwr
duration        nolimit
delay           0
*************** Bottom of Data *******
```

Figure 118. Example of /etc/u.map file

The MapName file contains the mapping between a subdirectory of a directory managed by automount and the mount parameters.

The MapName file can contain specific entries and a generic entry. There should be only one generic entry in a MapName file and it must be the first one. When the automounter tries to resolve a lookup request, it attempts to find a specific entry. If a specific entry does not exist for the name being looked up, it attempts to use the generic entry.

The variable <uc_name> means convert the name being looked up to upper case. Whenever this variable is encountered, it is replaced by the name being

looked up. A directory with the looked-up name is created and used as a mount point for the file system to be mounted. The <uc_name> can be used to replace any level qualifier in the data set. For example, if the name of the directory that is being looked up is user1, automount will resolve the name in the following ways:

OMVS.<uc_name> = OMVS.USER1
OMVS.<uc_name>.HFS = OMVS.USER1.HFS

The <uc_name> variable is replaced with the uppercase name anywhere in the string.

A.2.1.3 Step 3 - Start the automount facility

Figure 119 on page 252 contains an example of starting the automount facility (from the shell) and how file systems are automatically mounted. The automount command can only be issued from a **superuser ID**. It has the following syntax:

```
/usr/sbin/automount [-s] [Master filename]
```

When running the command with no arguments, the automount facility reads the /etc/auto.master file to determine the directories to be monitored and the file names that contain their configuration specifications, the MapName files. If automount is used with a master file name specified, that file name is used instead of /etc/auto.master.

The -s option only checks the syntax of the configuration file. The automount policy is not activated.

Figure 119 shows how <uc_name> works with the /etc/auto.master and /etc/u.map files from Figure 117 and Figure 118. HFS data sets, OMVS.SC64.USER1 and OMVS.SC64.USER2 have already been allocated. The low-level qualifier of the HFS data sets is the user ID, which is also the directory mount point that automount will dynamically allocate. With the automount facility, the HFS data set will be automatically mounted off the /u directory, as soon as a user tries to access any directory in their HFS file system. Type in OMVS from ISPF option 6 to access the UNIX System Services shell.

Important Information

When an HFS data set is first allocated for a new user (in our case, USER1 shown in Figure 116 on page 249), and automount is used to dynamically allocate a mount point, this new mount point directory's permission bits are set to 700 and the owner field is set to a superuser ID name. In order for USER1 to be able to use this new file system, you will need to issue the chown command to change the owner field of this new mount point directory that was created by automount. In Figure 119 on page 252, item 7 shows a superuser issuing the chown command against the /u/user1 directory to change the ownership.

```
IBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 1999
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

$ su
# df
Mounted on     Filesystem                Avail/Total    Files      Status
/var           (OMVS.SC64.VAR)           14104/14400    4294967276 Available
/etc           (OMVS.SC64.ETC)           84448/86400    4294967052 Available
/              (OMVS.SC64.ROOT)          158712/1368000 4294958129 Available
# /usr/sbin/automount         1
FOMF0107I Processing file /etc/u.map
FOMF0108I Managing directory /u 2
# df                          3
Mounted on     Filesystem                Avail/Total    Files      Status
/u             (*AMD/u)                  0/8            0          Available
/var           (OMVS.SC64.VAR)           14104/14400    4294967276 Available
/etc           (OMVS.SC64.ETC)           84448/86400    4294967052 Available
/              (OMVS.SC64.ROOT)          158712/1368000 4294958129 Available
# cd /u/user1                 4
# cd /u/user2                 4
# df                          5
Mounted on     Filesystem                Avail/Total    Files      Status
/u             (*AMD/u)                  0/8            0          Available
/u/user2       (OMVS.SC64.USER2)         1400/1440      4294967294 Available
/u/user1       (OMVS.SC64.USER1)         1400/1440      4294967294 Available
/var           (OMVS.SC64.VAR)           14104/14400    4294967276 Available
/etc           (OMVS.SC64.ETC)           84448/86400    4294967052 Available
/              (OMVS.SC64.ROOT)          158712/1368000 4294958129 Available
# ls -ldn /u/user1            6
drwx------   2 0        0          8192 Jun 25 02:20 /u/user1
# chown -R user1 /u/user1     7
# ls -ld /u/user1             8
drwx------   2 USER1    SYS1       8192 Jun 25 02:20 /u/user1
#
```

Figure 119. Result of using <uc_name> in MapName file

1 The automount command is issued from a superuser ID to start the automount facility from the UNIX System Services shell.

2 Automount scans the /etc/master.auto file first to see what MapName files should be read. Here we are managing the /u directory.

Note: Calling automount command twice erroneously does not create any problems, regardless of whether a file system is already mounted or not. Automount rereads the /etc/master.auto file and associated MapName files and picks up any changes.

3 The display free space command (df) is issued, showing that the automount facility is started and is monitoring the /u directory (*AMD/u).

4 Change directory (cd) commands are issued to access directories in two file systems to be mounted off the /u directory. In this case, the directories user1 and user2 are used to resolve the <uc_name> symbol in the /etc/u.map file. The user1 and user2 directory names are translated to uppercase and substituted to build the HFS data set names, OMVS.SC64.USER1 and OMVS.SC64.USER2, respectively.

The user1 and user2 directories do not physically exist in any file system, but they are created as pseudo-mount points by the automount facility on which the HFS data sets OMVS.SC64.USER1 and OMVS.SC64.USER2 are mounted.

5 Output from another display free space command (df) shows (*AMD/u) is monitoring the /u directory. It also shows the OMVS.SC64.USER1 and OMVS.SC64.USER2 data sets are now mounted at pseudo mount points /u/user1 and /u/user2 respectively.

Note: When automount is actively monitoring a particular mount point (in this case, /u), it is no longer possible to add a file to this directory (/u) or create a new subdirectory off the /u directory using the mkdir command. If attempted, you will receive the message:

```
EDC515I Dynamic allocation error
```

6 The ls -ldn /u/user1 command is issued against the /u/user1 directory and the directory attributes are displayed using the numeric values for UID and GID.

7 The chown command is issued to change the ownership of the /u/user1 directory from UID 0 to USER1.

8 The ls -ld /u/user1 command is issued again to show that the owner field of the /u/user1 directory is now set to USER1.

A.2.1.4 Step 4 - Starting automount when OMVS is started

When you have everything customized and working, you can have the automount facility started when the UNIX System Services kernel is started by adding it to the /etc/rc file. Add the following line to the /etc/rc file:

```
# Start the automount facility
/usr/sbin/automount
```

A.2.1.5 Step 5 - Using a specific entry in a mapname file

Use specific entries for directory names when the parameters you wish to use differ from the generic entry. Any parameters that are not specified are inherited from the generic entry. Figure 120 shows a specific entry defining a directory name called itsosj in the name parameter of the MapName file, rather than an *. Also notice that in this example the duration for generic mounts is set to unmount idle file systems after 60 minutes, but in our specific mount entry, idle file systems will stay mounted indefinitely.

```
BROWSE -- /etc/auto.master ------
COMMAND ===>
*************** Top of Data **********
/u                  /etc/u.map
*************** Bottom of Data *******

BROWSE -- /etc/u.map -------------
COMMAND ===>
*************** Top of Data **********
name            *
type            HFS
filesystem      OMVS.SC64.<uc_name>
mode            rdwr
duration        nolimit
delay           0
/*
name            itsosj
type            HFS
filesystem      OMVS.ITSOSJ.HFS
mode            rdwr
duration        60
delay           0
*************** Bottom of Data *******
```

Figure 120. Specific entry in a MapName file

Whenever the directory /u/itsosj is referenced by a command such as cd or cp, automount will mount HFS data set **OMVS.ITSOSJ.HFS**.

Important Information

- Do not use a slash (/) in front of the name of the directory to be mounted in a specific entry in a MapName file. For example, in /etc/u.map:

  ```
  name itsosj
  ```

  is correct, while

  ```
  name /itsosj
  ```

  is NOT correct.

- The directory name and the data set name qualifier for the HFS data set, which is replaced by the variable <uc_name>, have to be the same. Otherwise you will get error messages like:

  ```
  EDC129I No such file or directory
  ```

  or

  ```
  EDC515I Dynamic allocation error
  ```

A.2.2 Changing which data sets get automounted

The automount facility is really a physical file system (PFS) that is started with a FILESYSTYPE statement in the PARMLIB. After the PFS is started, automount manages the policy you make active through the /usr/sbin/automount command. You can change the automount policy at any time, although you cannot set it to null. To change it, update your automount configuration files and run the automount (/usr/sbin/automount).

A.2.3 Stopping the automount facility

You cannot stop automount after it has been started because there are no external commands to stop any PFS. To turn automount off, change your automount configuration files so that automount is set to manage a dummy directory. Then activate that configuration with the /usr/sbin/automount command.

Appendix B. Sample JCL and output

This appendix provides sample JCL that we have used in our project. Samples are provided in a logical order. We have also included additional information (such as ISPF Data Set Information, LISTCAT outputs and notes) to make it easier to follow the logical order, and to understand the associated results of the jobs.

B.1 HFS data set related information

There are several ways in which you can find information about HFS file systems.

B.1.1 D OMVS,F

You can use D OMVS,FILE to display a list of the file systems that UNIX System Services is currently using together with the status of each file system.

```
D OMVS,F
BPXO044I 18.25.50 DISPLAY OMVS 410
OMVS     000F ACTIVE          OMVS=(7B)
TYPENAME   DEVICE ----------STATUS----------- MODE QJOBNAME  QPID
TFS             5 ACTIVE                      RDWR
  NAME=/TMP
  PATH=/tmp
  MOUNT PARM=-s 500
HFS            12 ACTIVE                      RDWR
  NAME=OMVS.STYRES1.HFS3
  PATH=/u/guts
HFS             4 ACTIVE                      RDWR
  NAME=OMVS.SC63.VAR
  PATH=/var
HFS             3 ACTIVE                      RDWR
  NAME=OMVS.SC63.USERS
  PATH=/u
HFS             2 ACTIVE                      RDWR
  NAME=OMVS.SC63.ETC
  PATH=/etc
HFS             1 ACTIVE                      RDWR
  NAME=HFS.OS390R7.SC63.O37RA1.ROOT
  PATH=/
```

B.1.2 df

The df shell command gives you information about the space used and available within a file system. By default, it gives you space information in units of 512-byte blocks.

Note that the output of the df command may show different values from confighfs and ISPF. The df command also includes the space reserved for shadow writes of the metadata pages and so will usually show that more space is used. The values will be closest immediately after a sync operation.

Issued without parameters, it tells you about all mounted file systems.

```
STYRES3 @ SC63:/u/guts>df
Mounted on     Filesystem                Avail/Total    Files      Status
/tmp           (/TMP)                    999703/1000000 127962     Available
/u/guts        (OMVS.STYRES1.HFS3)       21208/21600    4294967284 Available
/var           (OMVS.SC63.VAR)           12904/12960    4294967293 Available
/u             (OMVS.SC63.USERS)         37216/37440    4294967282 Available
/etc           (OMVS.SC63.ETC)           73552/76320    4294967039 Available
/              (HFS.OS390R7.SC63.O37RA1.ROOT)203168/1379520 4294949032 Available
```

B.1.3 df -P

The -P parameter lists complete information about space used.

```
STYRES3 @ SC63:/>df -P
Filesystem         512-blocks       Used  Available  Capacity Mounted on
/TMP                  1000000        367     999633        1% /tmp
OMVS.STYRES1.HFS3       21600        392      21208        2% /u/guts
OMVS.SC63.VAR           12960         56      12904        1% /var
OMVS.SC63.USERS         37440        224      37216        1% /u
OMVS.SC63.ETC           76320       2768      73552        4% /etc
HFS.OS390R7.SC63.O37RA1.ROOT 1379520     1176352     203168       86% /
```

We have 1176352 blocks of 512 bytes. If we divide by eight, we get the number of pages which is 147044.

B.1.4 ISPF information for root HFS

ISPF/PDF can also tell us about the space used for an individual HFS data set.

Here, we see an example for the root data set.

```
Data Set Name  . . . : HFS.OS390R7.SC63.O37RA1.ROOT

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 958
 Storage class  . . . : OPENMVS        Allocated extents . : 2
  Volume serial . . . : SBOX14         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 147044
  Record length . . . : 0              % Utilized  . . . . : 85
  Block size  . . . . : 0              Number of members . : 18263
  1st extent cylinders: 878
  Secondary cylinders : 80
  Data set name type  : HFS
```

B.1.5 ISPF information for HFS OMVS.SC63.USERS

Following, we have a display from ISPF for a data set containing user file systems:

```
Data Set Name  . . . : OMVS.SC63.USERS

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 26
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX06         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 25
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 12
  1st extent cylinders: 26
  Secondary cylinders : 26
  Data set name type  : HFS
```

B.1.6 ISPF information for HFS OMVS.STYRES1.HFS3

```
Data Set Name  . . . : OMVS.STYRES1.HFS3

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 15
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX06         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 49
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 11
  1st extent cylinders: 15
  Secondary cylinders : 15
  Data set name type  : HFS
```

B.1.7 confighfs /

The next example shows the output from the confighfs command. This command was introduced by DFSMS/MVS 1.5.

For more detail on the confighfs command, please see 3.3.5, “confighfs shell command” on page 68.

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs /
Statistics for file system HFS.OS390R7.SC63.O37RA1.ROOT
( 06/18/99 6:36pm )
 File system size:____172440
                  _673.59375(MB)
 Used pages:      ____147044
                  _574.39063(MB)
 Attribute pages: ______1911
                  ____7.4648(MB)
 Cached pages:    _______167
                  ___0.65234(MB)
 Seq I/O reqs:    _________________499
 Random I/O reqs: __________________0
 Lookup hit:      ________________53697
 Lookup miss:     ________________17924
 1st page hit:    ________________10417
 1st page miss:   _________________430
 Index new tops:  __________________1
 Index splits:    _________________26
 Index joins:     __________________7
 Index read hit:  ______________102005
 Index read miss: _______________1363
 Index write hit: ________________10320
 Index write miss:__________________0
 RFS flags        _________________82(HEX)
 RFS error flags: __________________0(HEX)
 High format RFN: ________________272A2(HEX)
 Member count:    ___________41067008
 Sync interval:   _________________60(seconds)
```

B.1.7.1 HFRFN for OMVS.OS390R7.SC63.O37RC1.ROOT

The High Formatted page number is hexadecimal 272A2 or, in decimal, 160418. Dividing 160418 by 12 (the number of pages per track) gives us 13369 tracks used. Dividing again by 15 (the number of tracks per cylinder) gives us **892 formatted cylinders**.

B.1.8 configfs /u

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>configfs /u
Statistics for file system OMVS.SC63.USERS
( 06/17/99 7:36pm )
 File system size:_____4680
                  __18.28125(MB)
 Used pages:      _______25
                  0.09765625(MB)
 Attribute pages: _________4
                  __0.015625(MB)
 Cached pages:    _________2
                  _0.0078125(MB)
 Seq I/O reqs:    ___________________7
 Random I/O reqs: ___________________0
 Lookup hit:      __________________47
 Lookup miss:     __________________10
 1st page hit:    _________________654
 1st page miss:   __________________7
 Index new tops:  __________________1
 Index splits:    __________________0
 Index joins:     __________________0
 Index read hit:  __________________67
 Index read miss: __________________5
 Index write hit: __________________41
 Index write miss:__________________0
 RFS flags        _________________82(HEX)
 RFS error flags: _________________0(HEX)
 High format RFN: _________________18(HEX)
 Member count:    ________________6144
 Sync interval:   _________________60(seconds)
```

B.1.8.1 HFRFN for OMVS.SC63.USERS

The High Formatted page number is hexadecimal 18 or decimal 24.

As before, we can divide by 12 to gives us the number of tracks which is **2 formatted tracks.**

B.1.9 confighfs /u/guts output

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs /u/guts
Statistics for file system OMVS.STYRES1.HFS3
( 06/17/99 7:33pm )
 File system size:_____2700
                  _10.546875(MB)
 Used pages:      ________49
                  0.19140625(MB)
 Attribute pages: _________1
                  _0.0039063(MB)
 Cached pages:    _________4
                  __0.015625(MB)
 Seq I/O reqs:    ____________________1
 Random I/O reqs: ____________________0
 Lookup hit:      ____________________2
 Lookup miss:     ____________________1
 1st page hit:    ____________________3
 1st page miss:   ____________________5
 Index new tops:  ____________________0
 Index splits:    ____________________0
 Index joins:     ____________________0
 Index read hit:  ____________________3
 Index read miss: ____________________1
 Index write hit: ____________________0
 Index write miss:____________________0
 RFS flags        ___________________82(HEX)
 RFS error flags: ____________________0(HEX)
 High format RFN: ___________________31(HEX)
 Member count:    _______________12544
 Sync interval:   ___________________60(seconds)
```

B.1.9.1 HFRFN for OMVS.STYRES1.HFS3

The High Formatted page number is hexadecimal 31 or decimal 49. Dividing by 12 and rounding up gives us **5 formatted tracks.**

B.2 Sample DFSMSdss jobs

These are jobs that we used to test backup and recovery.

B.2.1 Logical DUMP without ALLDATA(*)

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.HFS3.DSSDUMP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(OMVS.STYRES1.HFS3)) -
      OUTDDNAME(OUT1)
/*
```

B.2.2 RESTORE with RENAMEU

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.STYRES1.HFS3.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    DATASET(INCLUDE(OMVS.STYRES1.HFS3)) -
    RENAMEU(OMVS.STYRES1.HFS3,OMVS.STYRES1.HFS3.BACKUP)
/*
```

B.2.2.1 ISPF Information for OMVS.STYRES1.HFS3.BACKUP

```
Data Set Name  . . . : OMVS.STYRES1.HFS3.BACKUP

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 1
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX15         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 49
  Record length . . . : 0              % Utilized  . . . . : 27
  Block size  . . . . : 0              Number of members . : 11
  1st extent cylinders: 1
  Secondary cylinders : 15
  Data set name type  : HFS
```

Note: The amount of allocated space was reduced from 15 cylinders to 1 cylinder, because the HFRFN pointed to the 5th track. The number of used pages was not changed.

B.2.3 Logical DUMP with ALLDATA(*)

Note: This HFS is not mounted.

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA),
//            SPACE=(CYL,(20,200)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.HFS1.DSSDUMP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(OMVS.STYRES1.HFS1)) -
      ALLDATA(*) -
      OUTDDNAME(OUT1)
/*
```

B.2.3.1 ISPF Information for HFS OMVS.STYRES1.HFS1

```
Data Set Name  . . . : OMVS.STYRES1.HFS1

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 30
 Storage class  . . . : OPENMVS        Allocated extents . : 2
  Volume serial . . . : SBOX13         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 157
  Record length . . . : 0              % Utilized  . . . . : 43
  Block size  . . . . : 0              Number of members . : 4
  1st extent tracks . : 15
  Secondary tracks  . : 15
  Data set name type  : HFS
```

B.2.4 RESTORE to a preallocated HFS with REPLACE

Note: We renamed the original HFS from OMVS.STYRES1.HFS1 to OMVS2.STYRES1.HFS2. Afterwards we allocated a new HFS with the same name as the original HFS by using ISPF option 3.2.

B.2.4.1 ISPF allocation panel (option 3.2)

```
 Allocate New Data Set
Command ===>
                                                                More:
Data Set Name  . . . : OMVS.STYRES1.HFS1

Management class . . .                (Blank for default management class)
Storage class  . . . .                (Blank for default storage class)
 Volume serial . . . .                (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . CYLINDER       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 50             (In above units)
 Secondary quantity    10             (In above units)
 Directory blocks  . . 0              (Zero for sequential data set) *
 Record format . . . . U
 Record length . . . . 0
 Block size  . . . . . 0
 Data set name type  : HFS            (LIBRARY, HFS, PDS, or blank)  *
```

Note: Our primary request of 50 cylinders was reduced to 42 cylinders of primary and 9 cylinders of secondary, due to the Default Device Geometry settings in ISMF. On our system the *Default Device Geometry* is set for a 3380 volume:

- Bytes/track . . . . . : 47476
- Tracks/cylinder . . . : 15

A 3390 volume has a capacity of 56664 bytes/track. Therefore, all allocations to a 3390 will be reduced by s factor of 1.19 (56664/47476).

B.2.4.2 RESTORE with REPLACE

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.STYRES1.HFS1.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    DATASET(INCLUDE(OMVS.STYRES1.HFS1)) -
    REPLACE
/*
```

B.2.4.3 ISPF Information for HFS OMVS.STYRES1.HFS1

```
 Data Set Information
Command ===>

Data Set Name  . . . : OMVS.STYRES1.HFS1

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 42
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX14         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 157
  Record length . . . : 0              % Utilized  . . . . : 2
  Block size  . . . . : 0              Number of members . : 4
  1st extent cylinders: 42
  Secondary cylinders : 9
  Data set name type  : HFS
```

Note: The ISPF Data Set Info shows that the HFS data set was increased from 30 tracks to 42 cylinders.

B.2.4.4 Dumping and filtering on DSORG

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(400,200)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(BY(DSORG,EQ,HFS)) -
      LOGINDYNAM(SBOX13) -
      OUTDDNAME(OUT1)
```

Note: We have specified TYPRUN=NORUN. Only input data set selection is done without actually processing data sets. Printed output for the run indicates the data sets selected.

B.2.4.5 Dumping and filtering on data set name

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(400,200)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(OMVS.**.HFS*.**)) -
      OUTDDNAME(OUT1)
```

B.2.4.6 Dumping with STORGRP parameter

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) -
      STORGRP(OPENMVS)) -
      OUTDDNAME(OUT1)
/*
```

B.2.4.7 Logical full volume dumps

Notes for the following samples:

1. We received a RC=4 (ADR377W for SYS1.VTOCIX.Vvolser and SYS1.VVDS.Vvolser data sets) for this job. This can be avoided by excluding these data sets from the operation using the EXCLUDE option such as:

   ```
   DATASET(INCLUDE(**) EXCLUDE(SYS1.VTOCIX.V**, SYS1.VVDS.**))
   ```

2. We have specified TYPRUN=NORUN. Only input data set selection is done without actually processing data sets. Printed output for the run indicates the data sets selected.

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) -
      LOGINDYNAM(SBOX13) -
      OUTDDNAME(OUT1)
/*
```

Note: We received a RC=8 with message ADR380E reason code 75 for every multi-volume HFS data set located on this volume. The reason code 75 indicates:

- All volumes of a multi-volume data set were not included in the input volume list and SELECTMULTI was not specified.

To bypass this problem, you can use *one* of the following jobs:

- Specifying all volumes that contain a part of a multi-volume data set:

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) -
      LOGINDYNAM((SBOX13),(SBOX14),(SBOX15),(SBOX16),(SBOX17)) -
      OUTDDNAME(OUT1)
/*
```

Note: We received an RC=4 (ADR377W for every SYS1.VTOCIX and SYS1.VVDS data set) on this job. This can be avoided by excluding these data sets from the operation using the EXCLUDE option such as:

```
DATASET(INCLUDE(**) EXCLUDE(SYS1.VTOCIX.V**, SYS1.VVDS.**))
```

- Specifying SELECTMULTI parameter:

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=327
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) -
      LOGINDYNAM(SBOX13) SELECTMULTI(ANY) -
      OUTDDNAME(OUT1)
/*
```

Note: We received an RC=4 (ADR377W for SYS1.VTOCIX.SBOX13 and SYS1.VVDS.VSBOX13) on this job. This can be avoided by excluding the index VTOC from the operation using the EXCLUDE option such as:

```
DATASET(INCLUDE(**) EXCLUDE(SYS1.VTOCIX.V**, SYS1.VVDS.**))
```

```
//DUMP     EXEC PGM=ADRDSSU,PARM=('TYPRUN=NORUN')
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=3276
//            DSN=OMVS.STYRES1.BACKUP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(**)) -
      LOGINDYNAM(SBOX13) SELECTMULTI(FIRST) -
      OUTDDNAME(OUT1)
/*
```

Note: We received an RC=4 (ADR377W for SYS1.VTOCIX.SBOX13 and SYS1.VVDS.VSBOX13) on this job. This can be avoided by excluding the index VTOC from the operation using the EXCLUDE option such as:

```
DATASET(INCLUDE(**) EXCLUDE(SYS1.VTOCIX.V**, SYS1.VVDS.**))
```

B.2.5 Physical DUMP

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA),
//            SPACE=(CYL,(20,20)),DCB=BLKSIZE=32760,
//            DSN=OMVS.STYRES1.HFS2.DSSDUMP.PHY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP INDYNAM(SBOX13) -
      DATASET(INCLUDE(OMVS.STYRES1.HFS2)) -
      OUTDDNAME(OUT1)
/*
```

Note: The HFS is not mounted.

Note: See B.2.3.1, “ISPF Information for HFS OMVS.STYRES1.HFS1” on page 264 for ISPF Data Set Information. We renamed the HFS in B.2.4.2, “RESTORE with REPLACE” on page 265 from OMVS.STYRES1.HFS1 to OMVS.STYRES.HFS2.

B.2.6 RESTORE (physical) with RENAMEU

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.STYRES1.HFS2.DSSDUMP.PHY,DISP=OLD
//OUT1     DD UNIT=3390,VOL=SER=SBOX13,
//            DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    OUTDD(OUT1) -
    DATASET(INCLUDE(OMVS.STYRES1.HFS2)) -
    RENAMEU(OMVS.STYRES1.HFS2,OMVS.STYRES.HFS2.BACKUP.PHY)
/*
```

B.2.6.1 ISPF Data Set Information

```
Data Set Name  . . . : OMVS.STYRES.HFS2.BACKUP.PHY

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 30
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX13         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 157
  Record length . . . : 0              % Utilized  . . . . : 43
  Block size  . . . . : 0              Number of members . : 4
  1st extent tracks . : 30
  Secondary tracks  . : 15
  Data set name type  : HFS
```

Note: The number of allocated tracks was not changed during physical restore processing.

B.3 DFSMSdss multi-volume processing

These are jobs that we used to test backup and recovery of multi-volume HFS data sets.

B.3.1 DUMP of root HFS

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA),
//            SPACE=(CYL,(800,200)),DCB=BLKSIZE=32760,
//            DSN=OMVS.SC63.ROOT.DSSDUMP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(HFS.OS390R7.SC63.O37RA1.ROOT)) -
      OUTDDNAME(OUT1)
/*
```

B.3.1.1 D OMVS,F

Note: The D OMVS,F command shows that the file system is quiesced during dump processing.

```
 D OMVS,F

HFS             4 ACTIVE                      RDWR
  NAME=OMVS.SC63.VAR
  PATH=/var
HFS             3 ACTIVE                      RDWR
  NAME=OMVS.SC63.USERS
  PATH=/u
HFS             2 ACTIVE                      RDWR
  NAME=OMVS.SC63.ETC
  PATH=/etc
HFS             1 QUIESCED                      RDWRSTYRES1D 1728053278
  NAME=HFS.OS390R7.SC63.O37RA1.ROOT
  PATH=/
```

B.3.1.2 LISTCAT ENT(OMVS.SC63.ROOT.BACKUP.MULTI) VOL

```
IDCAMS  SYSTEM SERVICES                                          TIM

/   LISTC ENT(HFS.OS390R7.SC63.O37RA1.ROOT) VOL
NONVSAM ------- HFS.OS390R7.SC63.O37RA1.ROOT
     IN-CAT --- UCAT.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.166
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX14      DEVTYPE------X'3010200F'
```

Note: The root file system is still a single file system.

B.3.2 RESTORE to multi-volume HFS data set using MAKEMULTI

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.SC63.ROOT.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    MAKEMULTI -
    DATASET(INCLUDE(HFS.OS390R7.SC63.O37RA1.ROOT)) -
    RENAMEU(HFS.OS390R7.SC63.O37RA1.ROOT,OMVS.SC63.ROOT.BACKUP.MULTI)
```

Note: The target HFS data set did not exist before the restore operation.

B.3.2.1 LISTCAT ENT(OMVS.SC63.ROOT.BACKUP.MULTI) VOL

```
IDCAMS  SYSTEM SERVICES                                    TIME: 18:50:43       06/18/99
1

/   LISTC ENT(OMVS.SC63.ROOT.BACKUP.MULTI) VOL                          00004001
NONVSAM ------- OMVS.SC63.ROOT.BACKUP.MULTI
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.169
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
VOLUMES
      VOLSER------------SBOX13     DEVTYPE------X'3010200F'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
      VOLSER------------------*     DEVTYPE------X'00000000'
```

Note: We truncated the output of the IDCAMS LISTCAT. The total number of candidate volumes was 59.

B.3.3 RESTORE to multi-volume HFS data set using VOLCOUNT(N((02))

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.SC63.ROOT.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    VOLCOUNT(N(02)) -
    DATASET(INCLUDE(HFS.OS390R7.SC63.O37RA1.ROOT)) -
    RENAMEU(HFS.OS390R7.SC63.O37RA1.ROOT,OMVS.SC63.ROOT.BACKUP.MULTI)
/*
```

Note: The target HFS data set did not exist before the restore operation.

B.3.3.1 LISTCAT ENT(OMVS.SC63.ROOT.BACKUP.MULTI) VOL

```
 COMMAND INPUT ===>                                              SCROLL ===>
/   LISTC ENT(OMVS.SC63.ROOT.BACKUP.MULTI) VOL                        000
NONVSAM ------- OMVS.SC63.ROOT.BACKUP.MULTI
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------1999.169
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS      MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX13      DEVTYPE------X'3010200F'
       VOLSER-----------------*      DEVTYPE------X'00000000'
```

Note that we now have one actual volume and one candidate volume shown.

B.3.4 RESTORE to multi-volume HFS data set using VOLCOUNT(ANY)

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.SC63.ROOT.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    VOLCOUNT(ANY) -
    DATASET(INCLUDE(HFS.OS390R7.SC63.O37RA1.ROOT)) -
    RENAMEU(HFS.OS390R7.SC63.O37RA1.ROOT,OMVS.SC63.ROOT.BACKUP.MULTI)
/*
```

Note: The target HFS data set did not exist before the restore operation.

B.3.4.1 ISPF Data Set Information

```
Data Set Name  . . . : OMVS.SC63.ROOT.BACKUP.MULTI1

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 892
 Storage class  . . . : OPENMVS        Allocated extents . : 5
  Volume serial . . . : SBOX15         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 147045
  Record length . . . : 0              % Utilized  . . . . : 91
  Block size  . . . . : 0              Number of members . : 18264
  1st extent cylinders: 327
  Secondary cylinders : 80
  Data set name type  : HFS
```

Note: The total amount of allocated space was reduced from 958 cylinders to 892 cylinders to match the HFRFN (see B.1.7, “confighfs /” on page 259).

B.3.4.2 LISTCAT ENT(OMVS.SC63.ROOT.BACKUP.MULTI1) VOL

```
IDCAMS  SYSTEM SERVICES

/   LISTC ENT(OMVS.SC63.ROOT.BACKUP.MULTI1) VOL
NONVSAM ------- OMVS.SC63.ROOT.BACKUP.MULTI1
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.169
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX15     DEVTYPE------X'3010200F'
       VOLSER------------SBOX16     DEVTYPE------X'3010200F'
       VOLSER------------SBOX14     DEVTYPE------X'3010200F'
```

Note: The HFS was converted from a single volume to a multi-volume data set (three volumes).

B.3.5 RESTORE to preallocated multi-volume HFS

Note: We cannot restore an HFS data set to a preallocated (target) HFS data set (requires REPLACE) with a different name (requires RENAME). Also, we could not restore to an HFS data set while it is mounted. Therefore, we need two DUMP and RESTORE processes.

B.3.5.1 DUMP

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,3),
//            SPACE=(CYL,(400,200)),DCB=BLKSIZE=32760,
//            DSN=OMVS.SC63.ROOT.BACKUP.MULTI1.DSSDUMP
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(OMVS.SC63.ROOT.BACKUP.MULTI1)) -
      OUTDDNAME(OUT1)
```

Note: Refer to B.3.4.1, “ISPF Data Set Information” on page 272 and to B.3.4.2, “LISTCAT ENT(OMVS.SC63.ROOT.BACKUP.MULTI1) VOL” on page 273 for more information about the allocation.

B.3.5.2 IDCAMS DELETE

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   DELETE (OMVS.SC63.ROOT.BACKUP.MULTI1)
/*
```

B.3.5.3 IEFBR14 to allocate a multi-volume HFS data set

```
//ALLOC    EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//DDNAME   DD DSN=OMVS.SC63.ROOT.BACKUP.MULTI1,
//            DISP=(NEW,KEEP),
//            UNIT=(SYSDA,5),
//            SPACE=(CYL,(1200,400,1)),
//            DSNTYPE=HFS
/*
```

Note: The number of directory blocks must be specified for an HFS data set but the value has no effect on allocation.

B.3.5.4 ISPF Information for the preallocated HFS (before RESTORE)

```
Data Set Name  . . . : OMVS.SC63.ROOT.BACKUP.MULTI1

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 1,006
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX16         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 5
  Record length . . . : 0              % Utilized  . . . . : 1
  Block size  . . . . : 0              Number of members . : 1
  1st extent cylinders: 1006
  Secondary cylinders : 336
  Data set name type  : HFS
```

Note: Our primary request of 1,200 cylinders was reduced to 1,006 cylinders of primary and 336 cylinders of secondary, due to the Default Device Geometry settings in ISMF. On our system the *Default Device Geometry* is set for a 3380 volume:

- Bytes/track . . . . . : 47476
- Tracks/cylinder . . . : 15

A 3390 volume has a capacity of 56664 bytes/track.Therefore, all allocations to a 3390 will be reduced by a factor of 1.19 (56664/47476).

B.3.5.5 RESTORE to preallocated HFS with REPLACE

```
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DSN=OMVS.SC63.ROOT.BACKUP.MULTI1.DSSDUMP,DISP=OLD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    REPLACE -
    DATASET(INCLUDE(OMVS.SC63.ROOT.BACKUP.MULTI1))
```

B.3.5.6 ISPF information for the preallocated HFS (after RESTORE)

```
 Data Set Name  . . . : OMVS.SC63.ROOT.BACKUP.MULTI1

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 1,006
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX16         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 147045
  Record length . . . : 0              % Utilized  . . . . : 81
  Block size  . . . . : 0              Number of members . : 18264
  1st extent cylinders: 1006
  Secondary cylinders : 336
  Data set name type  : HFS
```

Note: The number of used pages does not change. But now we have increased our primary allocation from 892 cylinders to 1,006 cylinders.

B.3.5.7 LISTCAT for the preallocated HFS (before and after RESTORE)

```
IDCAMS  SYSTEM SERVICES

/   LISTC ENT(OMVS.SC63.ROOT.BACKUP.MULTI1) VOL
NONVSAM ------- OMVS.SC63.ROOT.BACKUP.MULTI1
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.169
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX16     DEVTYPE------X'3010200F'
       VOLSER------------------*     DEVTYPE------X'00000000'
       VOLSER------------------*     DEVTYPE------X'00000000'
       VOLSER------------------*     DEVTYPE------X'00000000'
       VOLSER------------------*     DEVTYPE------X'00000000'
```

Note: The number of volumes was increased to 5. Four of these are candidate volumes.

B.4 Increasing the size of an HFS data set

This section shows how we increased the size of an HFS data set:

- confighfs -x extends an HFS file system.
- confighfs -xn extends an HFS file system to a new volume.

B.4.1 Using confighfs -x

B.4.1.1 ISPF Information

```
Data Set Name  . . . : OMVS.STYRES1.HFS4

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 3
 Storage class  . . . : OPENMVS        Allocated extents . : 1
  Volume serial . . . : SBOX17         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 5
  Record length . . . : 0              % Utilized  . . . . : 13
  Block size  . . . . : 0              Number of members . : 1
  1st extent tracks . : 3
  Secondary tracks  . : 0
  Data set name type  : HFS
```

Note: At this time, the primary allocation is three tracks and no secondary extents.

B.4.1.2 confighfs -x /u/styres1

```
STYRES3 @ SC63:/>df
Mounted on     Filesystem               Avail/Total    Files      Status
/u/styres1     (OMVS.STYRES1.HFS4)      128/288        4294967292 Available

STYRES3 @ SC63:/>cd u/styres1
STYRES3 @ SC63:/u/styres1>cp file1 file3
cp: FSUM6259 target file "file3": EDC5133I No space left on device.
STYRES3 @ SC63:/u/styres1>

STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs -x 2T /u/styres1
STYRES3 @ SC63:/usr/lpp/dfsms/bin>df
Mounted on     Filesystem               Avail/Total    Files      Status
/u/styres1     (OMVS.STYRES1.HFS4)      320/480        4294967292 Available
```

Notes:

1. The copy of file1 to file3 has failed.
2. Therefore we increased the size of the HFS data set 'OMVS.STRYRES.HFS4' by 2 tracks.
3. The number of total blocks was increased from 3 tracks (3 track * 12 page/track * 8 block/page = 288 (512byte) blocks) to 5 tracks (480 blocks).

B.4.1.3 ISPF Information after confighfs

```
Data Set Name  . . . : OMVS.STYRES1.HFS4

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 5
 Storage class  . . . : OPENMVS        Allocated extents . : 2
  Volume serial . . . : SBOX17         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 20
  Record length . . . : 0              % Utilized  . . . . : 33
  Block size  . . . . : 0              Number of members . : 3
  1st extent tracks . : 3
  Secondary tracks  . : 0
  Data set name type  : HFS
```

Note: Now, we have a second extent and the current allocation is five tracks. However, we still have no secondary extents specified (secondary tracks = 0).

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>cd /u/styres1
STYRES3 @ SC63:/u/styres1>ls
file1  file2
STYRES3 @ SC63:/u/styres1>cp file1 file3
```

Note: Now, we can successfully copy the file.

```
STYRES3 @ SC63:/u/styres1>cp file1 file4
STYRES3 @ SC63:/u/styres1>cp file1 file5
cp: FSUM6259 target file "file5": EDC5133I No space left on device.
STYRES3 @ SC63:/u/styres1>
```

Note: The confighfs has not changed the secondary allocation value. So the HFS cannot extend automatically. Therefore, we can run into the same situation again.

B.4.2 Using confighfs -xn and IDCAMS ALTER ADDVOLUMES

B.4.2.1 confighfs -xn

```
 STYRES3 @ SC63:/u/styres1>cd /usr/lpp/dfsms/bin
STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs -xn 4T /u/styres1
Error issuing PFSCTL: RC=0 ERRNO=133(85) REASON=5B27C005
ERRNO=133(85): No space available
```

Note: The confighfs -xn to extend the HFS to another volume fails if no candidate volumes are available for this HFS data set. See the next screen.

B.4.2.2 LISTCAT VOLUME

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   LISTC ENT(OMVS.STYRES1.HFS4) VOL
/*
```

The LISTCAT joblog shows:

```
IDCAMS  SYSTEM SERVICES

   LISTC ENT(OMVS.STYRES1.HFS4) VOL
NONVSAM ------- OMVS.STYRES1.HFS4
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------1999.173
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS      MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX17      DEVTYPE------X'3010200F'
```

Note: The HFS data set 'OMVS.STYRES1.HFS4' is a single volume data set.

B.4.2.3 IDCAMS ALTER ADDVOLUMES

Note: We unmounted the HFS data set before we ran the IDCAMS ALTER:

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
   ALTER OMVS.STYRES1.HFS4 ADDVOLUMES(* *)
/*
```

Note: This job adds two candidate volumes to the catalog entry for the HFS data set 'OMVS.STRES1.HFS4'.

```
IDCAMS  SYSTEM SERVICES

   LISTC ENT(OMVS.STYRES1.HFS4) VOL
NONVSAM ------- OMVS.STYRES1.HFS4
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------1999.173
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS      MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX17      DEVTYPE------X'3010200F'
       VOLSER------------------*      DEVTYPE------X'00000000'
       VOLSER------------------*      DEVTYPE------X'00000000'
```

Note: The IDCAMS LISTCAT now shows the two candidate volumes.

B.4.2.4 HFS automatically extended to the next volume

```
STYRES3 @ SC63:/usr/lpp/dfsms/bin>df
Mounted on     Filesystem               Avail/Total    Files      Status
/u/styres1     (OMVS.STYRES1.HFS4)      128/480        4294967290 Available

STYRES3 @ SC63:/usr/lpp/dfsms/bin>cd /u/styres1
STYRES3 @ SC63:/u/styres1>cp file1 file5
STYRES3 @ SC63:/u/styres1>df
Mounted on     Filesystem               Avail/Total    Files      Status
/u/styres1     (OMVS.STYRES1.HFS4)      608/960        4294967290 Available
```

Notes:

1. We have mounted the HFS data set again.
2. The HFS data set was extended automatically to the next volume.
3. Now, the total number of allocated 512-byte blocks is 960 (10 tracks).

B.4.2.5 ISPF information

```
Data Set Name  . . . : OMVS.STYRES1.HFS4

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 10
 Storage class  . . . : OPENMVS        Allocated extents . : 3
  Volume serial . . . : SBOX17         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 56
  Record length . . . : 0              % Utilized  . . . . : 46
  Block size  . . . . : 0              Number of members . : 6
  1st extent tracks . : 3
  Secondary tracks  . : 0
  Data set name type  : HFS
```

B.4.2.6 LISTCAT VOLUME

```
IDCAMS  SYSTEM SERVICES

    LISTC ENT(OMVS.STYRES1.HFS4) VOL
NONVSAM ------- OMVS.STYRES1.HFS4
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)     CREATION--------1999.173
       RELEASE----------------2     EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS     MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX17     DEVTYPE------X'3010200F'
       VOLSER------------SBOX13     DEVTYPE------X'3010200F'
       VOLSER-----------------*     DEVTYPE------X'00000000'
```

Note: The LISTCAT output shows that the HFS data set was extended to the next volume.

B.4.2.7 confighfs -xn 4T

```
STYRES3 @ SC63:/u/styres1>cd /usr/lpp/dfsms/bin
STYRES3 @ SC63:/usr/lpp/dfsms/bin>confighfs -xn 4T /u/styres1
STYRES3 @ SC63:/usr/lpp/dfsms/bin>df
Mounted on     Filesystem              Avail/Total    Files      Status
/u/styres1     (OMVS.STYRES1.HFS4)     896/1344       4294967289 Available
```

Notes:

1. We issued a second confighfs -xn command to extend the HFS to the third volume.
2. The total number of allocated 512-byte blocks was increased from 960 (10 tracks) to 1344 (14 tracks).

B.4.2.8 ISPF information

```
Data Set Name  . . . : OMVS.STYRES1.HFS4

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated tracks  . : 14
 Storage class  . . . : OPENMVS        Allocated extents . : 4
  Volume serial . . . : SBOX17         Maximum dir. blocks : NOLIMIT
  Device type . . . . : 3390
 Data class . . . . . :
  Organization  . . . : PO            Current Utilization
  Record format . . . : U              Used pages  . . . . : 56
  Record length . . . : 0              % Utilized  . . . . : 33
  Block size  . . . . : 0              Number of members . : 6
  1st extent tracks . : 3
  Secondary tracks  . : 0
  Data set name type  : HFS
```

Note: The *Current Allocation* is 14 tracks on four extents. We still have **no** secondary allocation specified in the format 1 DSCB in the VTOC.

B.4.2.9 LISTCAT VOLUME

```
IDCAMS  SYSTEM SERVICES

    LISTC ENT(OMVS.STYRES1.HFS4) VOL
NONVSAM ------- OMVS.STYRES1.HFS4
     IN-CAT --- MCAT.SANDBOX.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------1999.173
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS ----OPENMVS      MANAGEMENTCLASS---MCDB22
       DATACLASS --------(NULL)      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX17      DEVTYPE------X'3010200F'
       VOLSER------------SBOX13      DEVTYPE------X'3010200F'
       VOLSER------------SBOX15      DEVTYPE------X'3010200F'
```

Note: The LISTCAT VOLUME output shows that the fourth extent went onto the third volume.

B.5 Shared HFS - DFSMSdss dump from client

We tested sysplex sharing support in a sysplex consisting of three systems: SC63, SC64 and SC65.

B.5.1 Mounting the HFS on system SC65

Using TSO MOUNT command in a batch job.

```
//STEP1 EXEC PGM=IKJEFT01
//SYSPRINT  DD  SYSOUT=*
//SYSTSPRT  DD  SYSOUT=*
//SYSTSIN  DD *
  MOUNT FILESYSTEM('NIGELR2.TEST.HFS')    +
        MOUNTPOINT('/u/nigelr2')          +
        TYPE(HFS)  MODE(RDWR)             +
        PARM('SYNC(120),NOWRITEPROTECT')  +
        SYSNAME(SC65)                     +
        NOAUTOMOVE
/*
```

B.5.2 Displaying the ownership

Using D OMVS,F system command.

```
D OMVS,F
BPXO045I 15.08.28 DISPLAY OMVS 655
OMVS     000F ACTIVE          OMVS=(9B)
TYPENAME   DEVICE ----------STATUS----------- MODE
...
HFS            18 ACTIVE                     RDWR
  NAME=NIGELR2.TEST.HFS
  PATH=/u/nigelr2
  MOUNT PARM=SYNC(120),NOWRITEPROTECT
  OWNER=SC65     AUTOMOVE=N CLIENT=Y
...
```

B.5.3 Displaying the ENQ

Displaying all ENQ's for an HFS data set.

```
D GRS,RES=(*,NIGELR2.TEST.HFS)
ISG343I 15.09.48 GRS STATUS 657
S=SYSTEMS SYSDSN   NIGELR2.TEST.HFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      OMVS                 000F     008FDE28   SHARE      OWN
S=SYSTEMS SYSZDSN  NIGELR2.TEST.HFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      OMVS                 000F     008F77B8 EXCLUSIVE    OWN
```

Note: Only the owning system SC65 (server) holds the ENQ's. System SC64 (client) is part of the SYSBPX group. SC64 doesn't held any ENQ for this particular HFS data set, but it can also access the HFS in read-write mode.

B.5.4 Performing the DFSMSdss logical dump

DFSMSdss logical dump from the client system SC64.

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,KEEP),
//            UNIT=(SYSDA,6),STORCLAS=SCMHLRES,
//            SPACE=(CYL,(200,200)),DCB=BLKSIZE=32760,
//            DSN=NIGELR2.TEST.HFS.DSSDUMP.#2
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(NIGELR2.TEST.HFS)) -
      OUTDDNAME(OUT1)
/*
```

Note: You can also specify CONCURRENT which specifies that the data is to be processed with concurrent copy if possible.

B.5.4.1 Displaying the status during dump processing

The D OMVS,F system command shows that the HFS is quiesced during the logical dump processing.

```
D OMVS,F
BPXO045I 15.59.51 DISPLAY OMVS 672
OMVS     000F ACTIVE          OMVS=(9B)
TYPENAME   DEVICE ----------STATUS----------- MODE
...
HFS            18 QUIESCED                 RDWR
  NAME=NIGELR2.TEST.HFS
  PATH=/u/nigelr2
  MOUNT PARM=SYNC(120),NOWRITEPROTECT
  OWNER=SC65     AUTOMOVE=N CLIENT=Y
  QSYSTEM=SC64 QJOBNAME=NIGELR2D QPID=  50462741
...
```

The df USS command returns an error message during dump processing because the HFS data set is quiesced.

```
NIGELR2 @ SC64:/>df
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/tmp      (/SC65/TMP)               999742/1000000 127967     Available
/SC64/tmp      (/SC64/TMP)               999719/1000000 127964     Available
FSUMF168 w_getmntent could not obtain mount point for "NIGELR2.TEST.HFS".
/SC65/var      (HFS.SC65.VAR)            74768/77760    4294967022 Available
/SC65/etc      (HFS.SC65.ETC)            48952/51840    4294967044 Available
/SC65/dev      (HFS.SC65.DEV)            14224/14400    4294967286 Available
/SC65          (WTSCPLX2.SC65.SYSTEM.HFS)992/1440       4294967275 Available
/was/hod       (HFS.SC64.WAS.HOD)        42568/64800    4294966422 Available
/web/hod       (HFS.SC64.WEB.HOD)        52400/57600    4294966750 Available
/SC64/var      (HFS.SC64.VAR)            36776/38880    4294967277 Available
/SC64/etc      (HFS.SC64.ETC)            85088/87840    4294967055 Available
/SC64/dev      (HFS.SC64.DEV)            14224/14400    4294967283 Available
/O39RA1        (HFS.OS390R9.O39RA1.ROOT) 2674040/2675520 4294938691 Available
/SC64          (WTSCPLX2.SC64.SYSTEM.HFS)1032/1440      4294967279 Available
/              (WTSCPLX2.SYSPLEX.ROOT)   0/1536         4294967271 Available
```

B.5.4.2 Displaying the status after dump processing

Afterwards, if the file system is unquiesced, the df -v USS command once more shows the information about the file system (including the owner).

```
NIGELR2 @ SC64:/>df -v
Mounted on     Filesystem               Avail/Total    Files      Status
...
/u/nigelr2     (NIGELR2.TEST.HFS)       262032/2888640 4294938692 Available
HFS, Read/Write
SYNC(120),NOWRITEPROTECT
File System Owner : SC65
...
```

B.6 Recovery samples

This section shows how we tested a recovery procedure. To do this, we mounted a file system and then broke the HFS data set internal structure.

B.6.1 Corrupt metadata recovery (example 1)

We mount our file system on mount point */u/nigelr2/old* and display the file structure.

```
NIGELR5 @ SC65:/>df
Mounted on     Filesystem               Avail/Total    Files      Status
...
/u/nigelr2/old (NIGELR2.TEST.HFS1.BAD2)  256/1440       4294967264 Available
...
NIGELR5 @ SC65:/>cd /u/nigelr2/old
NIGELR5 @ SC65:/u/nigelr2/old>ls
dir2    dir3    dir5    file1   file10  file11  file12  file13  file14
NIGELR5 @ SC65:/u/nigelr2/old>cd dir3/dir4
NIGELR5 @ SC65:/u/nigelr2/old/dir3/dir4>ls
dir50    dir6     file400  file401  file402  file403  file404  file41   file42
NIGELR5 @ SC65:/u/nigelr2/old/dir3/dir4>cd dir50
NIGELR5 @ SC65:/u/nigelr2/old/dir3/dir4/dir50>ls
file500     file501     file502     symlink503
```

We start a TSM backup for this file system.

```
dsmc> sel /u/nigelr2/old/ -sub=y
Selective Backup function invoked.

Directory-->              8,192 /u/nigelr2/old/ [Sent]
Directory-->              8,192 /u/nigelr2/old/dir2 [Sent]
Directory-->              8,192 /u/nigelr2/old/dir3 [Sent]
Directory-->              8,192 /u/nigelr2/old/dir5 [Sent]
Normal File-->            1,427 /u/nigelr2/old/file1 [Sent]
.
. (output lines removed for clarity)
.
Directory-->              8,192 /u/nigelr2/old/dir5/dir51 [Sent]
Normal File-->            2,800 /u/nigelr2/old/dir5/file51 [Sent]
Normal File-->            7,682 /u/nigelr2/old/dir5/file52 [Sent]
Normal File-->            1,440 /u/nigelr2/old/dir5/dir51/file51a
[Sent]
Symbolic Link-->             18 /u/nigelr2/old/dir5/dir51/symlink
[Sent]
Selective Backup processing of '/u/nigelr2/old/' finished without failure.


Total number of objects inspected:       31
Total number of objects backed up:       31
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:          0
Total number of objects failed:           0
Total number of bytes transferred:   331.47 KB
Data transfer time:                    1.70 sec
Network data transfer rate:          194.89 KB/sec
Aggregate data transfer rate:         50.36 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:00:06
```

Now, we break the file system by zapping a specific metadata page on DASD volume.

We try to display the attributes for a path which we broke before by using the `ls -l` USS command.

```
NIGELR2 @ SC64:/>cd u/nigelr2/old
NIGELR2 @ SC64:/u/nigelr2/old>ls
dir2    dir3    dir5    file1   file10  file11  file12  file13  file14
NIGELR2 @ SC64:/u/nigelr2/old>cd dir3
NIGELR2 @ SC64:/u/nigelr2/old/dir3>ls
dir4    file31
NIGELR2 @ SC64:/u/nigelr2/old/dir3>cd dir4
NIGELR2 @ SC64:/u/nigelr2/old/dir3/dir4>ls
dir50    dir6     file400  file401  file402  file403  file404  file41   file42
NIGELR2 @ SC64:/u/nigelr2/old/dir3/dir4>ls -l
ls: ./dir50: EDC5157I An internal error has occurred.
ls: ./dir6: EDC5157I An internal error has occurred.
ls: ./file404: EDC5157I An internal error has occurred.
total 208
-r-xr-xr-x   1 NIGELR2  SYS1        1350 Mar 29 11:41 file400
-r-xr-xr-x   1 NIGELR2  SYS1       13350 Mar 29 11:41 file401
-r-xr-xr-x   1 NIGELR2  SYS1       66690 Mar 29 11:41 file402
-r-xr-xr-x   1 NIGELR2  SYS1        3000 Mar 29 11:41 file403
-r-xr-xr-x   1 NIGELR2  SYS1        6467 Mar 28 17:33 file41
-r-xr-xr-x   1 NIGELR2  SYS1        2581 Mar 28 17:34 file42
NIGELR2 @ SC64:/u/nigelr2/old/dir3/dir4>
```

Note: The file *file404* and the directories *dir50* and *dir6* are displayed on a *ls* command, but not on a *ls -l* command because the *ls -l* command needs to read the attributes (which are not available).

At this stage, a TSM backup is also impossible due to its inability to traverse the damaged tree structure. The error message, which does not look helpful at first glance, is also a pointer to the problem we have caused within the HFS:

```
dsmc> sel /u/nigelr2/old/ -sub=y
Node Name: WTSC64OE
Session established with server ADSM: MVS
  Server Version 3, Release 1, Level 2.40
  Server date/time: 03/29/2000 16:46:31  Last access: 03/29/2000 13:44:37

Selective Backup function invoked.

ANS1028S Internal program error.  Please see your service representative.

Total number of objects inspected:       13
Total number of objects backed up:        0
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:          0
Total number of objects failed:           0
Total number of bytes transferred:        0
Data transfer time:                    0.00 sec
Network data transfer rate:            0.00 KB/sec
Aggregate data transfer rate:          0.00 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:00:01
```

To start to fix this problem, we need to mount a new file system on a different mount point */u/nigelr2/new*.

```
NIGELR2 @ SC64:/>df
Mounted on     Filesystem                Avail/Total    Files      Status
...
/u/nigelr2/old (NIGELR2.TEST.HFS1.BAD2)  10336/11520    4294967264 Available
/u/nigelr2/new (NIGELR2.TEST.HFS1.RECOVER.#2)13592/14400    4294967269 Availabl
...
```

The copytree (using the check tree function) reports some errors within the file system.

```
NNIGELR2 @ SC64:/u>copytree /u/nigelr2/old
Checking /u/nigelr2/old
Scanning for file nodes...
Skipping mountpoint: /u/nigelr2/old/..
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir50
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir6
Processing 26 nodes
Cannot open file: /u/nigelr2/old//dir3/dir4/file404

******************

Check complete.  Error count= 3
Directory errors:   2
File errors:        1
Symlink errors:     0
Char-spec errors:   0
FIFO errors:        0
Sparse file count:  0
NIGELR2 @ SC64:/u>
```

Now we can copy the "good" files and directories by using copytree into the new file system at mount point */u/nigelr2/new/*.

Note the errors from copytree, trying to access two complete directories and one file. We will need TSM to help us with these later.

```
NIGELR2 @ SC64:/u>copytree /u/nigelr2/old /u/nigelr2/new
Copying /u/nigelr2/old to /u/nigelr2/new
Scanning for file nodes...
Skipping mountpoint: /u/nigelr2/old/..
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir50
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir6
Processing 26 nodes
Creating directories
Creating other files
Cannot open file: /u/nigelr2/old//dir3/dir4/file404
Setting file attributes

******************

Copy complete.  Error count= 3
Directory errors:   2
Directories copied: 7
File errors:        1
Files copied:       17
Symlink errors:     0
Symlinks copied:    1
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
NIGELR2 @ SC64:/u>
```

Alternatively, you can use the pax USS command to copy the remaining files and directories into the new file system at mount point */u/nigelr2/new/.*

```
NIGELR2 @ SC64:/u/nigelr2/old>pax -r -w . /u/nigelr2/new/
NIGELR2 @ SC64:/u/nigelr2/old>cd ..
NIGELR2 @ SC64:/u/nigelr2>cd new
NIGELR2 @ SC64:/u/nigelr2/new>ls
dir2    dir3    dir5    file1   file10  file11  file12  file13  file14
NIGELR2 @ SC64:/u/nigelr2/new>cd dir3
NIGELR2 @ SC64:/u/nigelr2/new/dir3>cd dir3
cd: dir3: EDC5129I No such file or directory.
NIGELR2 @ SC64:/u/nigelr2/new/dir3>cd dir4
NIGELR2 @ SC64:/u/nigelr2/new/dir3/dir4>ls -l
total 208
-r-xr-xr-x   1 NIGELR2  SYS1        1350 Mar 29 11:41 file400
-r-xr-xr-x   1 NIGELR2  SYS1       13350 Mar 29 11:41 file401
-r-xr-xr-x   1 NIGELR2  SYS1       66690 Mar 29 11:41 file402
-r-xr-xr-x   1 NIGELR2  SYS1        3000 Mar 29 11:41 file403
-r-xr-xr-x   1 NIGELR2  SYS1        6467 Mar 28 17:33 file41
-r-xr-xr-x   1 NIGELR2  SYS1        2581 Mar 28 17:34 file42
NIGELR2 @ SC64:/u/nigelr2/new/dir3/dir4>
```

Now we can restore only the missing files and directories to our new file system at mount point */u/nigel/new* by using the TSM restore function. The -ifnewer command line option ensures that we do not overwrite the files that were undamaged by our file system corruption. These files may be newer than the last backup version, and we do not want to make matters worse by overwriting them!

```
dsmc> res /u/nigelr2/old/ /u/nigelr2/new/ -sub=y -ifnewer
Session established with server ADSM: MVS
  Server Version 3, Release 1, Level 2.40
  Server date/time: 03/29/2000 18:12:00  Last access: 03/29/2000 18:01:57

Restore function invoked.

ANS1247I Waiting for files from the server...
Restoring           8,192 /u/nigelr2/old/ --> /u/nigelr2/new/ [Done]
Restoring           8,192 /u/nigelr2/old/dir2 --> /u/nigelr2/new/dir2
[Done]
Restoring           8,192 /u/nigelr2/old/dir3 --> /u/nigelr2/new/dir3
[Done]
Restoring           8,192 /u/nigelr2/old/dir5 --> /u/nigelr2/new/dir5
[Done]
Restoring           8,192 /u/nigelr2/old/dir3/dir4 --> /u/nigelr2/new/dir3/dir4
Restoring           8,192 /u/nigelr2/old/dir3/dir4/dir50 --> /u/nigelr2/new/dir3
Restoring           8,192 /u/nigelr2/old/dir3/dir4/dir6 --> /u/nigelr2/new/dir3/
Restoring           8,192 /u/nigelr2/old/dir3/dir4/dir60 --> /u/nigelr2/new/dir3
Restoring           8,192 /u/nigelr2/old/dir5/dir51 --> /u/nigelr2/new/dir5/dir5
File /u/nigelr2/new/file1 exists, skipping
File /u/nigelr2/new/file10 exists, skipping
File /u/nigelr2/new/file11 exists, skipping
File /u/nigelr2/new/file12 exists, skipping
File /u/nigelr2/new/file13 exists, skipping
File /u/nigelr2/new/file14 exists, skipping
File /u/nigelr2/new/dir2/file3 exists, skipping
File /u/nigelr2/new/dir3/file31 exists, skipping
File /u/nigelr2/new/dir3/dir4/file400 exists, skipping
File /u/nigelr2/new/dir3/dir4/file401 exists, skipping
File /u/nigelr2/new/dir3/dir4/file402 exists, skipping
File /u/nigelr2/new/dir3/dir4/file403 exists, skipping
Restoring           3,026 /u/nigelr2/old/dir3/dir4/file404 --> /u/nigelr2/new/di
File /u/nigelr2/new/dir3/dir4/file41 exists, skipping
File /u/nigelr2/new/dir3/dir4/file42 exists, skipping
Restoring           1,960 /u/nigelr2/old/dir3/dir4/dir50/file500 --> /u/nigelr2/
Restoring           3,432 /u/nigelr2/old/dir3/dir4/dir50/file501 --> /u/nigelr2/
Restoring         177,800 /u/nigelr2/old/dir3/dir4/dir50/file502 --> /u/nigelr2/
Restoring              18 /u/nigelr2/old/dir3/dir4/dir50/symlink503 --> /u/nigel
Restoring           1,360 /u/nigelr2/old/dir3/dir4/dir60/file600 --> /u/nigelr2/
File /u/nigelr2/new/dir5/file51 exists, skipping
File /u/nigelr2/new/dir5/file52 exists, skipping
File /u/nigelr2/new/dir5/dir51/file51a exists, skipping
File /u/nigelr2/new/dir5/dir51/symlink exists, skipping

Restore processing finished.

Total number of objects restored:        15
Total number of objects failed:           0
Total number of bytes transferred:   183.29 KB
Data transfer time:                    0.30 sec
Network data transfer rate:          606.88 KB/sec
Aggregate data transfer rate:         51.73 KB/sec
Elapsed processing time:           00:00:03
```

Note: Now, we have recovered the broken file system (HFS data set). But the restored files and directories will only be current as of the date and time of the last TSM backup. So we need to verify if the restored files and directories have been changed since the last TSM backup (if possible!).

Please keep in mind that no error flag is set in PFS for this file system. You can verify the error flag by using confighfs command (must be run on **server** system).

```
NIGELR5 @ SC65:/>cd /usr/lpp/dfsms/bin
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.TEST.HFS1.BAD2
( 03/29/00 5:58pm )
 File system size:_____1080
                  ___4.21875(MB)
 Used pages:      _______117
                  0.45703125(MB)
 Attribute pages: _________6
                  _0.0234375(MB)
 Cached pages:    ________34
                  _0.1328125(MB)
 Seq I/O reqs:    ___________________18
 Random I/O reqs: ___________________0
 Lookup hit:      ___________________74
 Lookup miss:     ___________________46
 1st page hit:    ___________________85
 1st page miss:   ___________________18
 Index new tops:  ___________________0
 Index splits:    ___________________0
 Index joins:     ___________________0
 Index read hit:  __________________251
 Index read miss: ___________________6
 Index write hit: ___________________72
 Index write miss:___________________0
 RFS flags        ___________________82(HEX)
 RFS error flags: _____________0(HEX)
 High foramt RFN: ___________________75(HEX)
 Member count:    ___________________31
 Sync interval:   ___________________60(seconds)
```

The same confighfs command fails on a client system.

```
NIGELR2 @ SC64:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Error issuing PFSCTL: RC=0 ERRNO=129(81) REASON=5B360105
ERRNO=129(81): HFS is not mounted
NIGELR2 @ SC64:/O39RA1/usr/lpp/dfsms/bin>
```

B.6.2 Corrupt metadata recovery (example 2)

We use the same 'broken' HFS data set as used in the first sample. But now, we created a **new file** in the affected file system at mount point */u/nigelr2/old*. The new file cannot be written to DASD at sync time, because the PFS detects the corrupted metadata structure during sync processing. The PFS sets an error flag in the main control block (RFS) for the affected file system. To reset the RFS error flag, you must unmount and mount the file system (HFS data set) again.

```
NIGELR5 @ SC65:/u>cd /usr/lpp/dfsms/bin
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.TEST.HFS1.BAD2
( 03/30/00 11:41am )
 File system size:_____1440
                  _____5.625(MB)
 Used pages:      _______117
                  0.45703125(MB)
 Attribute pages: _________6
                  _0.0234375(MB)
 Cached pages:    _________0
                  _________0(MB)
 Seq I/O reqs:    ___________________0
 Random I/O reqs: ___________________0
 Lookup hit:      ___________________0
 Lookup miss:     ___________________0
 1st page hit:    ___________________0
 1st page miss:   ___________________0
 Index new tops:  ___________________0
 Index splits:    ___________________0
 Index joins:     ___________________0
 Index read hit:  ___________________7
 Index read miss: ___________________0
 Index write hit: ___________________0
 Index write miss:___________________0
 RFS flags        _________________82(HEX)
 RFS error flags: _____________________0(HEX)
 High foramt RFN: _________________75(HEX)
 Member count:    _________________31
 Sync interval:   _________________60(seconds)
```

We create a new file */u/nigelr2/old/output* in the affected file system.

```
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>cd /u/nigelr2/old
NIGELR5 @ SC65:/u/nigelr2/old>ls -l >>output
```

Since the sync has not occurred, the error flag is not set.

```
NIGELR5 @ SC65:/u/nigelr2/old>cd /usr/lpp/dfsms/bin
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.TEST.HFS1.BAD2
( 03/30/00 11:42am )
 File system size:_____1440
                  _____5.625(MB)
 Used pages:      _______118
                  _0.4609375(MB)
 Attribute pages: _________6
                  _0.0234375(MB)
 Cached pages:    _________0
                  _________0(MB)
...
 Index write miss:___________________0
 RFS flags        ___________________8E(HEX)
 RFS error flags: ___________________0(HEX)
 High foramt RFN: ___________________75(HEX)
 Member count:    ___________________32
 Sync interval:   ___________________60(seconds)
```

A couple seconds later, the sync processing was performed. After the sync, the error flag is **on**.

```
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.TEST.HFS1.BAD2
( 03/30/00 11:42am )
 File system size:_____1440
                  _____5.625(MB)
 Used pages:      _______118
                  _0.4609375(MB)
 Attribute pages: _________6
                  _0.0234375(MB)
 Cached pages:    _________0
                  _________0(MB)
...
Index write miss:___________________0
 RFS flags        ___________________8E(HEX)
 RFS error flags: ___________________80(HEX)
 High foramt RFN: ___________________75(HEX)
 Member count:    ___________________32
 Sync interval:   ___________________60(seconds)
```

If the error flag is on, a copytree (checktree) fails.

```
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>cd /u
NIGELR5 @ SC65:/u>copytree /u/nigelr2/old
Checking /u/nigelr2/old
Scanning for file nodes...
Cannot read directory: /u/nigelr2/old
Processing 0 nodes

******************

Check complete.  Error count= 1
Directory errors:   1
File errors:        0
Symlink errors:     0
Char-spec errors:   0
FIFO errors:        0
Sparse file count:  0
NIGELR5 @ SC65:/u>
```

We also cannot copy the files and directories by using copytree.

```
NIGELR5 @ SC65:/u>copytree /u/nigelr2/old /u/nigelr2/new
Copying /u/nigelr2/old to /u/nigelr2/new
Scanning for file nodes...
Cannot read directory: /u/nigelr2/old
Processing 0 nodes
Creating directories
Creating other files
Setting file attributes

******************

Copy complete.  Error count= 1
Directory errors:   1
Directories copied: 0
File errors:        0
Files copied:       0
Symlink errors:     0
Symlinks copied:    0
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
NIGELR5 @ SC65:/u>
```

Also pax does not copy anything (no files in target file system).

```
NIGELR5 @ SC65:/u>cd /u/nigelr2/old
NIGELR5 @ SC65:/u/nigelr2/old>pax -r -w . /u/nigelr2/new
NIGELR5 @ SC65:/u/nigelr2/old>cd ../new
NIGELR5 @ SC65:/u/nigelr2/new>ls
NIGELR5 @ SC65:/u/nigelr2/new>
```

A df command shows that the affected file system is mounted in read-write mode.

```
NIGELR5 @ SC65:/u/nigelr2/new>df -v
Mounted on     Filesystem                Avail/Total    Files      Status
...
/u/nigelr2/new (NIGELR2.TEST.HFS1.RECOVER.#3)14224/14400    4294967294 Available
HFS, Read/Write
File System Owner : SC65
/u/nigelr2/old (NIGELR2.TEST.HFS1.BAD2)  10304/11520    4294967263 Available
HFS, Read/Write
File System Owner : SC65
/SC65/var      (HFS.SC65.VAR)            74752/77760    4294967020 Available
...
```

To reset the error flag, we **unmount and then mount the file system** again. We mount the file system in read-only mode at the same mount point.

```
NIGELR5 @ SC65:/u/nigelr2/new>df -v
Mounted on     Filesystem                Avail/Total    Files      Status
...
/u/nigelr2/new (NIGELR2.TEST.HFS1.RECOVER.#3)14224/14400    4294967294 Available
HFS, Read/Write
File System Owner : SC65
/u/nigelr2/old (NIGELR2.TEST.HFS1.BAD2)  11272/11520    4294967264 Available
HFS, Read Only
File System Owner : SC65
...
```

Now, the RFS error flag is reset.

```
NIGELR5 @ SC65:/u/nigelr2/new>cd /usr/lpp/dfsms/bin
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.TEST.HFS1.BAD2
( 03/30/00 12:40pm )
...
 RFS flags          _____________________82(HEX)
 RFS error flags: _____________________0(HEX)
 High foramt RFN: _____________________75(HEX)
 Member count:    _____________________31
 Sync interval:   _____________________60(seconds)
```

Now, the checktree function of copytree tool detects the invalid metadata structure.

```
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>cd /u
NIGELR5 @ SC65:/u>copytree /u/nigelr2/old
Checking /u/nigelr2/old
Scanning for file nodes...
Skipping mountpoint: /u/nigelr2/old/..
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir50
Cannot read directory: /u/nigelr2/old/dir3/dir4/dir6
Processing 26 nodes
Cannot open file: /u/nigelr2/old//dir3/dir4/file404

******************

Check complete.  Error count= 3
Directory errors:   2
File errors:        1
Symlink errors:     0
Char-spec errors:   0
FIFO errors:        0
Sparse file count:  0
NIGELR5 @ SC65:/u>
```

We can also copy the remaining files and directories to our new file system.

```
NIGELR5 @ SC65:/u>cd nigelr2/old
NIGELR5 @ SC65:/u/nigelr2/old>pax -r -w . /u/nigelr2/new
NIGELR5 @ SC65:/u/nigelr2/old>cd ../new
NIGELR5 @ SC65:/u/nigelr2/new>ls -l
total 120
drwxr-xr-x   2 NIGELR5  SYS1        8192 Mar 24 19:45 dir2
drwxr-xr-x   3 NIGELR5  SYS1        8192 Mar 28 17:34 dir3
drwxr-xr-x   3 NIGELR5  SYS1        8192 Mar 28 18:16 dir5
-rw-r--r--   1 NIGELR5  SYS1        1427 Mar 24 19:44 file1
-r-xr-xr-x   1 NIGELR5  SYS1        2057 Mar 29 14:13 file10
-r-xr-xr-x   1 NIGELR5  SYS1         238 Mar 28 17:32 file11
-r-xr-xr-x   1 NIGELR5  SYS1       15561 Mar 28 18:14 file12
-r-xr-xr-x   1 NIGELR5  SYS1         392 Mar 28 18:15 file13
-r-xr-xr-x   1 NIGELR5  SYS1        3115 Mar 28 18:15 file14
NIGELR5 @ SC65:/u/nigelr2/new>cd dir3/dir4
NIGELR5 @ SC65:/u/nigelr2/new/dir3/dir4>ls -l
total 208
-r-xr-xr-x   1 NIGELR5  SYS1        1350 Mar 29 11:41 file400
-r-xr-xr-x   1 NIGELR5  SYS1       13350 Mar 29 11:41 file401
-r-xr-xr-x   1 NIGELR5  SYS1       66690 Mar 29 11:41 file402
-r-xr-xr-x   1 NIGELR5  SYS1        3000 Mar 29 11:41 file403
-r-xr-xr-x   1 NIGELR5  SYS1        6467 Mar 28 17:33 file41
-r-xr-xr-x   1 NIGELR5  SYS1        2581 Mar 28 17:34 file42
NIGELR5 @ SC65:/u/nigelr2/new/dir3/dir4>
```

Note: The file *file404*, the directories *dir50* and *dir6* and the files behind the directories could not copied due to the broken index structure.

Next, we must restore the missing files to our new file system by using a TSM restore. See B.6.1, “Corrupt metadata recovery (example 1)” on page 284 for an example of TSM restore processing.

B.6.3 Lost volume recovery (example 3)

In this sample. we used a multi-volume HFS data set. We want to simulate the loss of a volume in this sample.

We created a multi-volume HFS data set spanning three volumes, as shown on the following LISTCAT report.

```
NONVSAM ------- NIGELR2.MULTIVOL.HFS
     IN-CAT --- UCAT.VSBOX01
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------2000.095
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE--------------HFS
     SMSDATA
       STORAGECLASS -----SCTEST      MANAGEMENTCLASS---MCDB22
       DATACLASS -----------HFS      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER------------SBOX27      DEVTYPE------X'3010200F'
       VOLSER------------SBOX25      DEVTYPE------X'3010200F'
       VOLSER------------SBOX26      DEVTYPE------X'3010200F'
READY
```

The file system is mounted on the same mount point as used in the samples before.

TSO ISHELL - file system attributes shows:

```
File System Attributes

File system name:
NIGELR2.MULTIVOL.HFS
Mount point:
/u/nigelr2/old


Status . . . . . : Available
File system type . . . : HFS
Mount mode . . . . . : R/W
Device number  . . . . : 112
Type number  . . . . . : 1
DD name  . . . . . : SYS00073
Block size . . . . . : 4096
Total blocks . . . . . : 1080
Available blocks . . . : 50
Blocks in use  . . . . : 999
```

We dumped our HFS data set by using the following job:

```
//DUMP     EXEC PGM=ADRDSSU
//OUT1     DD DISP=(NEW,CATLG),
//            STORCLAS=OPENMVS,
//            SPACE=(CYL,(5,5)),DCB=BLKSIZE=32760,
//            DSN=NIGELR2.MULTIVOL.HFS.DSSDUMP.ALLDATA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DUMP DATASET(INCLUDE(NIGELR2.MULTIVOL.HFS)) -
      ALLDATA(*) -
      OUTDDNAME(OUT1)
/*
```

We also backed up the entire file system (HFS data set) by using TSM selective backup.

```
dsmc> sel /u/nigelr2/old/ -sub=y
Selective Backup function invoked.

Directory-->               8,192 /u/nigelr2/old/ [Sent]
Directory-->               8,192 /u/nigelr2/old/dir001 [Sent]
Normal File-->           520,000 /u/nigelr2/old/file001 [Sent]
Normal File-->           240,000 /u/nigelr2/old/file002 [Sent]
Directory-->               8,192 /u/nigelr2/old/dir001/dir002 [Sent]
Directory-->               8,192 /u/nigelr2/old/dir001/dir003 [Sent]
Normal File-->           679,966 /u/nigelr2/old/dir001/file003 [Sent]
Normal File-->           899,940 /u/nigelr2/old/dir001/file004 [Sent]
Directory-->               8,192 /u/nigelr2/old/dir001/dir003/dir004
[Sent]
Normal File-->           450,000 /u/nigelr2/old/dir001/dir003/dir004/file006
[Sent]
Normal File-->           450,000 /u/nigelr2/old/dir001/dir003/dir004/file007
[Sent]
Normal File-->           530,000 /u/nigelr2/old/dir001/dir003/dir004/file008
[Sent]
Normal File-->           195,580 /u/nigelr2/old/dir001/dir003/dir004/file009
[Sent]
Normal File-->            44,000 /u/nigelr2/old/dir001/dir003/dir004/file010
[Sent]
Selective Backup processing of '/u/nigelr2/old/' finished without failure.


Total number of objects inspected:       14
Total number of objects backed up:       14
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:          0
Total number of objects failed:           0
Total number of bytes transferred:     3.82 MB
Data transfer time:                    7.79 sec
Network data transfer rate:          502.21 KB/sec
Aggregate data transfer rate:        264.39 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:00:14
```

Now, we:

- Create a new file: dir001/dir003/dir004/file101.

- Create a directory: dir101
- Create a symbolic link: dir101/sym001 (pointing to /usr/lpp/dfsms/bin)
- Delete a file: file010
- Modify file file001 (as shown on the following screen):

```
EDIT       /u/nigelr2/old/file001
 Command ===>
 ****** ***************************************************** Top
 000001
 000002   - modified after TSM backup and DSS dump
 000003
 000004 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 000005 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 000006 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 ...
```

Now, we run an incremental TSM backup, which picks up the two changed files. Note that it also generates a number of "expiring" messages on our system, some of which are shown in the screen shot. This is because the previous backup structure, for the corruption test, contained a number of directories that are no longer present in our new multi-volume structure. By default, TSM would not restore these, but the backup copies still exist in an "inactive" state. If we felt we did still need these files, the inactive files could be restored. Note that inactive files are also managed using a different TSM policy than active files.

Also note the long elapsed time, even though we are only backing up two files. TSM still needs to traverse the tree and compare the file information to the server database. We could have reduced this time by using the incrbydate option (see 6.4.1, “Tuning” on page 164).

```
dsmc> i /u/nigelr2/old/ -sub=y

Incremental backup of volume '/u/nigelr2/old/'
Expiring-->               8,192 /u/nigelr2/old/... [Sent]
Expiring-->              98,304 /u/nigelr2/old/bin/IBM/FSUMVCHA [Sent]
Expiring-->               1,475 /u/nigelr2/old/bin/IBM/FSUMURUN [Sent]
Expiring-->             360,448 /u/nigelr2/old/bin/IBM/FSUMSYMN [Sent]
...
Directory-->              8,192 /u/nigelr2/old/dir101 [Sent]
...
Expiring-->                  12 /u/nigelr2/old/dev [Sent]
Normal File-->          520,047 /u/nigelr2/old/file001 [Sent]
Expiring-->                  16 /u/nigelr2/old/krb5 [Sent]
Expiring-->               4,803 /u/nigelr2/old/newfile1 [Sent]
Expiring-->                 632 /u/nigelr2/old/newfile11 [Sent]
Expiring-->                  12 /u/nigelr2/old/tmp [Sent]
Expiring-->                  12 /u/nigelr2/old/var [Sent]
Expiring-->              44,000 /u/nigelr2/old/dir001/dir003/dir004/file010
[Sent]
Normal File-->           10,000 /u/nigelr2/old/dir001/dir003/dir004/file101
[Sent]
Symbolic Link-->             18 /u/nigelr2/old/dir101/sym001 [Sent]
Successful incremental backup of '/u/nigelr2/old/'


Total number of objects inspected:       32
Total number of objects backed up:        4
Total number of objects updated:          0
Total number of objects rebound:          0
Total number of objects deleted:     54,201
Total number of objects failed:           0
Total number of bytes transferred:   517.73 KB
Data transfer time:                    0.00 sec
Network data transfer rate:        563,366.04 KB/sec
Aggregate data transfer rate:          0.33 KB/sec
Objects compressed by:                    0%
Elapsed processing time:           00:25:32
```

We again modify file *file001* as shown in the following screen.

```
EDIT       /u/nigelr2/old/file001
Command ===>
****** ***************************************************** Top
000001
000002   - modified after TSM backup and DSS dump
000003
000004   - modified a second time after TSM incremental backup
000005
000006  /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
000007  /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
000008  /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
...
```

Next, we want to **simulate a hardware failure**. We did this by setting offline a volume (SBOX25) in the middle of our HFS file system. Afterwards, we cannot access any data (files or directories).

```
NIGELR2 @ SC64:/u/nigelr2/old>ls
NIGELR2 @ SC64:/u/nigelr2/old>ls -l
total 0
NIGELR2 @ SC64:/u/nigelr2/old>
```

The confighfs output shows that the error flag was set for our affected file system.

```
NIGELR5 @ SC65:/O39RA1/usr/lpp/dfsms/bin>confighfs /u/nigelr2/old
Statistics for file system NIGELR2.MULTIVOL.HFS
( 04/05/00 1:28pm )
 File system size:_____1080
                  ___4.21875(MB)
 Used pages:      _______994
                  _3.8828125(MB)
 Attribute pages: _________5
                  0.01953125(MB)
 Cached pages:    _________0
                  _________0(MB)
 Seq I/O reqs:    ___________________38
 Random I/O reqs: ___________________0
 Lookup hit:      __________________125
 Lookup miss:     ___________________0
 1st page hit:    __________________70
 1st page miss:   __________________39
 Index new tops:  ___________________1
 Index splits:    ___________________0
 Index joins:     ___________________0
 Index read hit:  __________________196
 Index read miss: ___________________1
 Index write hit: __________________120
 Index write miss:___________________0
 RFS flags        __________________82(HEX)
 RFS error flags: ___________40(HEX)
 High foramt RFN: __________________418(HEX)
 Member count:    __________________17
 Sync interval:   __________________60(seconds)
```

The checktree function of copytree utility also reports a 'Directory error'.

```
NIGELR2 @ SC64:/u>copytree /u/nigelr2/old
Checking /u/nigelr2/old
Scanning for file nodes...
Cannot read directory: /u/nigelr2/old
Processing 0 nodes

******************

Check complete.  Error count= 1
Directory errors:    1
File errors:         0
Symlink errors:      0
Char-spec errors:    0
FIFO errors:         0
Sparse file count:   0
```

We allocated a new HFS data set 'NIGELR2.MULTIVOL.HFS.NEW' and mounted this at mount point /u/nigelr2/new.

```
NIGELR2 @ SC64:/u>df
Mounted on     Filesystem                Avail/Total    Files      Status
...
/u/nigelr2/old (NIGELR2.MULTIVOL.HFS)    368/8640       4294967278 Available
/u/nigelr2/new (NIGELR2.MULTIVOL.HFS.NEW)9904/10080     4294967294 Available
...
```

Now, we tried to copy the affected file system into the new file system (HFS data set) to salvage as many files and directories as we could. However, it wasn't possible.

```
NIGELR2 @ SC64:/u>copytree /u/nigelr2/old /u/nigelr2/new
Copying /u/nigelr2/old to /u/nigelr2/new
Scanning for file nodes...
Cannot read directory: /u/nigelr2/old
Processing 0 nodes
Creating directories
Creating other files
Setting file attributes

******************

Copy complete.  Error count= 1
Directory errors:   1
Directories copied: 0
File errors:        0
Files copied:       0
Symlink errors:     0
Symlinks copied:    0
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
```

We also tried to unmount and then to remount the affected HFS data set in read-only mode again. However, this was not possible either. We got a dynamic allocation error during mount processing.

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
 Mount a File System

 Mount point:
                                                                  More:
      /u/nigelr2/old

    File system name . . 'NIGELR2.MULTIVOL.HFS
    File system type . . HFS
    New owner  . . . . . ________
    Owning system  . . . ________

    Select additional mount options:
    _ Read-only file system
    _  Ignore SETUID and SETGID
    _  Bypass security
    _  Do not automove file system
    Mount parameter:

     F1=Help       F3=Exit       F4=Name       F6=Keyshelp  F12=Cancel

 Errno=97x Dynamic allocation error; Reason=02180000 The return code value
 describes the error.  Press Enter to continue.
```

Therefore, we need to start our recovery from scratch. This means that we must first restore from our DFSMSdss dump. Then, we will use TSM RESTORE (with option ifnewer) to restore the file system to the time of the last TSM backup.

We used this DFSMSdss restore job (including an IDCAMS delete step so that we could restore to a new name):

```
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
/   DELETE (NIGELR2.MULTIVOL.HFS.RESTORE)
/*
//RESTORE  EXEC PGM=ADRDSSU
//IN1      DD  DISP=OLD,
//           DSN=NIGELR2.MULTIVOL.HFS.DSSDUMP.ALLDATA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 RESTORE INDD(IN1) -
    STORCLAS(SCTEST) -
    VOLCOUNT(N(3)) -
    DATASET(INCLUDE(NIGELR2.MULTIVOL.HFS)) -
    RENAMEU(NIGELR2.MULTIVOL.HFS,NIGELR2.MULTIVOL.HFS.RESTORE)
/*
```

We mounted the restored HFS data set on mount point /u/nigelr2/new:

```
NIGELR2 @ SC64:/u>tso ish
NIGELR2 @ SC64:/u>df
Mounted on     Filesystem                Avail/Total    Files      Status
...
/u/nigelr2/new (NIGELR2.MULTIVOL.HFS.RESTORE)400/8640       4294967281 Available
...
```

As you can see, we restored *file001*. However, this is not the latest version since *file001* was modified after the logical dump and then modified again after the TSM backup.

```
 BROWSE -- /u/nigelr2/new/file001 --------------------
 Command ===>
******************************** Top of Data ********
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
 /u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
```

We issued the TSM restore command to restore all files that were backed up to file systems mounted at /u/nigelr2/new. We also specified parameter ifnewer to restore only these files and directories that were backed up after the logical dump was taken.

```
dsmc> res /u/nigelr2/old/* /u/nigelr2/new/ -sub=y -ifnewer
Restore function invoked.

ANS1247I Waiting for files from the server...
Restoring           8,192 /u/nigelr2/old/ --> /u/nigelr2/new/ [Done]
Restoring           8,192 /u/nigelr2/old/dir001 --> /u/nigelr2/new/dir001
[Done]
Restoring           8,192 /u/nigelr2/old/dir101 --> /u/nigelr2/new/dir101
[Done]
Restoring           8,192 /u/nigelr2/old/dir001/dir002 --> /u/nigelr2/new/dir001
/dir002 [Done]
Restoring           8,192 /u/nigelr2/old/dir001/dir003 --> /u/nigelr2/new/dir001
/dir003 [Done]
Restoring           8,192 /u/nigelr2/old/dir001/dir003/dir004 --> /u/nigelr2/new
/dir001/dir003/dir004 [Done]


--- User Action is Required ---
File '/u/nigelr2/new/file001' exists

Select an appropriate action
  1. Replace this object
  2. Replace all objects that already exist
  3. Skip this object
  4. Skip all objects that already exist
  A. Abort this operation
Action [1,2,3,4,A] : 1
Restoring         520,047 /u/nigelr2/old/file001 --> /u/nigelr2/new/file001
[Done]
File /u/nigelr2/new/file002 exists, skipping
File /u/nigelr2/new/dir001/file003 exists, skipping
File /u/nigelr2/new/dir001/file004 exists, skipping
File /u/nigelr2/new/dir001/dir003/dir004/file006 exists, skipping
File /u/nigelr2/new/dir001/dir003/dir004/file007 exists, skipping
File /u/nigelr2/new/dir001/dir003/dir004/file008 exists, skipping
File /u/nigelr2/new/dir001/dir003/dir004/file009 exists, skipping
Restoring          10,000 /u/nigelr2/old/dir001/dir003/dir004/file101 --> /u/nig
elr2/new/dir001/dir003/dir004/file101 [Done]
Restoring              18 /u/nigelr2/old/dir101/sym001 --> /u/nigelr2/new/dir101
/sym001 [Done]

Restore processing finished.

Total number of objects restored:         9
Total number of objects failed:           0
Total number of bytes transferred:   517.71 KB
Data transfer time:                    0.62 sec
Network data transfer rate:          828.73 KB/sec
Aggregate data transfer rate:         22.84 KB/sec
Elapsed processing time:           00:00:22
```

The TSM restore brought back the directory *dir101*, the symbolic link *sym001* and the file *file001*, all of which were modified after the logical dump.

```
BROWSE -- /u/nigelr2/new/file001 ------------------
Command ===>
****************************** Top of Data ******

 - modified after TSM backup and DSS dump

/u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
/u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
/u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
/u/nigelr2/old/file001 in hfs nigelr2.multivol.hfs
...
```

However, note that the version of *file001* that was restored is not the most recent as that was modified *after* the TSM backup but before the hardware failure was simulated.

Also note, that *file010* was brought back by the DFSMSdss RESTORE job. It was deleted after the dump but before the TSM backup. However, it still remains as the TSM RESTORE does not delete files.

```
NIGELR2 @ SC64:/u/nigelr2/new>ls -l
total 1520
drwxrwxrwx   4 STC      SYS1        8192 Apr  4 18:05 dir001
drwxrwxrwx   2 STC      SYS1        8192 Apr  5 14:22 dir101
-rwxrwxrwx   1 STC      SYS1      520047 Apr  5 11:30 file001
-rwxrwxrwx   1 STC      SYS1      240000 Apr  4 18:03 file002
NIGELR2 @ SC64:/u/nigelr2/new>cd dir001/dir003/dir004
NIGELR2 @ SC64:/u/nigelr2/new/dir001/dir003/dir004>ls -l
total 3296
-rwxrwxrwx   1 STC      SYS1      450000 Apr  4 18:05 file006
-rwxrwxrwx   1 STC      SYS1      450000 Apr  4 18:06 file007
-rwxrwxrwx   1 STC      SYS1      530000 Apr  4 18:06 file008
-rwxrwxrwx   1 STC      SYS1      195580 Apr  4 18:07 file009
-rwxrwxrwx   1 STC      SYS1       44000 Apr  4 18:08 file010
-rwxrwxrwx   1 STC      SYS1       10000 Apr  5 11:27 file101
NIGELR2 @ SC64:/u/nigelr2/new/dir001/dir003/dir004>
```

The final step is to decide what to do about *file010* and *file001*. Almost certainly, you would simply choose to delete *file010*. For *file001*, it is unlikely in practice that you would know that it had been changed. We only know this because we have been checking what has happened in detail. The action that you would probably take is to check the date and time of the last TSM backup and notify users that changes made after then have been lost.

Appendix C. Special notices

This publication is intended to help storage administrators to implement and manage HFS data sets. The information in this publication is not intended as the specification of any programming interfaces that are provided by DFSMS/MVS V1R5. See the PUBLICATIONS section of the IBM Programming Announcement for DFSMS/MVS V1R5 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DFSORT	RETAIN
RMF	S/390
DFSMS	DFSMS/MVS
DFSMSdfp	DFSMSdss
DFSMShsm	Websphere
Hiperspace	IBM ®
IMS	MVS
MVS/DFP	MVS/ESA
Parallel Sysplex	OpenEdition
System/390	OS/390
RAMAC	SP
RACF	RAMAC

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 313.

- *OS/390 Version 2 Release 6 UNIX System Services Implementation and Customization,* SG24-5178
- *S/390 File and Print Sharing,* SG24-5330
- *Lotus Domino for S/390 Release 5: Problem Determination Guide*, SG24-5599
- *Lotus Domino for S/390 Release 5: Installation, Customization and Administration,* SG24-2083
- *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, SG24-5268
- *Implementing Concurrent Copy*, GG24-3990

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *OS/390 Version 2 Release 9 UNIX System Services Planning,* SC28-1890
- *OS/390 Version 2 Release 9 UNIX System Services User’s Guide,* SC28-1891
- *OS/390 Version 2 Release 9 UNIX System Services File System Interface Reference,* SC28-1909
- *OS/390 Version 2 Release 9 UNIX System Services Command Reference,* SC28-1892
- *OS/390 Version 2 Release 9 MVS Initialization and Tuning Reference,* SC28-1752

- *OS/390 Version 2 Release 9 TSO/E Command Reference,* SC28-1969
- *OS/390 Version 2 Release 9 UNIX System Services Assembler Callable Services,* SC28-1899
- *OS/390 Version 2 Release 8 Security Server Command Language Reference,* SC28-1919
- *OS/390 Version 2 Release 6 Network File System User's Guide,* SC26-7254
- *DFSMS/MVS Version 1 Release 5 Using Data Sets,* SC26-4922
- *DFSMS/MVS Version 1 Release 5 Planning for Installation,* SC26-4919
- *DFSMS/MVS Version 1 Release 5 Implementing System-Managed Storage,* SC26-3123
- *DFSMS/MVS Version 1 Release 5 DFSMSdss Storage Administration Reference,* SC26-4929
- *DFSMS/MVS Version 1 Release 5 DFSMSdss Storage Administration Guide,* SC26-4930
- *DFSMS/MVS Version 1 Release 5 DFSMSdfp Storage Administration Reference,* SC26-4920
- *DFSMS/MVS Version 1 Release 5 Access Method Services for the Integrated Catalog Facility,* SC26-4906
- *Storage Management Library: Managing Storage Groups,* SC26-3125
- *Storage Management Library: Managing Data,* SC26-3124
- *Using 3390 in an MVS Environment,* GC26-4574
- *DFSMS/MVS Version 1 Release 5 DFSMShsm Implementation and Customization*, SH21-1078
- *DFSMS/MVS Version 1 Release 5 DFSMShsm Managing Your Own Data*, SH21-1077
- *DFSMS/MVS Version 1 Release 5 DFSMShsm Storage Administration Guide*, SH21-1076
- *DFSMS/MVS Version 1 Release 5 DFSMShsm Storage Administration Reference*, SH21-1075
- *DFSMS/MVS Macro Instructions for Data Sets,* SC26-4913
- *OS/390 Version 2 Release 9 MVS System Commands*, GC28-1781
- *Tivoli Storage Manager for UNIX: Using the Backup-Archive Clients,* SH26-4105
- *Program Directory for the Tivoli Storage Manager Backup-Archive Client,* GI10-4520
- *OS/390 Version 2 Release 9 MVS Setting Up a Sysplex,* GC28-1779

D.4 Referenced Web sites

These Web sites are relevant as information sources:

- `http://www.tivoli.com/tsm` contains the latest information about the Tivoli Storage Manager.
- `http://www.s390.ibm.com/oe` and `http://www.s390.ibm.com/unix` both take you to the OS/390 UNIX System Services Web site. This site has a great deal of information about HFS and you can download the `copytree` utility from here.
- `http://www.ibm.com/support/techdocs` contains technical support flashes.

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of abbreviations

ACDS	active control data set
ACS	automatic class selection
AD	attribute directory
ADSM	adstar distributed storage manager
APAR	authorized program analysis report
APF	authorized programming facility
API	application programming interface
ASCII	American national standard for information interchange
BSAM	basic sequential access method
CC	concurrent copy
COMMDS	communication data set
DADSM	direct access storage space management
DASD	direct access storage device
DC	data class
DCB	data control block
DFSMS	data facility storage mangement subsystem
DFSMSdfp	data facility storage mangement subsystem data facility product
DFSMSdss	data facility storage mangement subsystem data set service
DFSMShsm	data facility storage mangement subsystem hierarchical storage management
DOS	disk operating system
DSCB	data set control block
ESDS	entry sequenced data set
FIFO	first in first out
FMID	function modification identifier
FPM	fragmented parcel map
FSN	file sequence number
GID	group identifier
GRS	global resource sharing
HFRFN	high formatted relative file number
HFS	hierarchical file system
HLQ	high level qualifier
IOS	input output supervisor
IPL	initial program load
ISMF	interactive storage management facility
ISPF	interactive structured programming facility
ITSO	International Technical Support Organization
JCL	job control language
KSDS	key sequenced data set
LFS	logical file system
LLQ	low level qualifier
MC	management class
ML1	migration level 1
ML2	migration level 2
MSR	millisecond response
MVS	multiple virtual storage
ND	name directory
NFS	network file system
PDSE	partitioned data set extended
PFS	physical file system
PTF	program temporary fix
QSAM	queued sequential access method
RACF	resource access control facility
RMF	resource measurement facility
RSM	real storage manager
RVA	RAMAC Virtual Array
SC	storage class
SCDS	source control data set
SG	storage group
SML	storage management library
SMS	storage management subsystem
SRM	system resource manager
SVC	supervisor call
TCB	task control block
TCP/IP	transmittion control protocol/internet protocol

TFS	temporary file system
TSM	Tivoli storage manager
UCB	unit control block
UID	user identifier
USS	UNIX system services
VCC	virtual concurrent copy
VSAM	virtual storage access method
VTOC	volume table of contents
WLM	workload manager

Index

D

E

F

G

H

I

J

K

Q

R

S

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5482-01 Hierarchical File System Usage Guide
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

Hierarchical File System Usage Guide

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Hierarchical File System Usage Guide

Redbooks

Learn how to manage HFS data

Optimize HFS performance

Set up sysplex sharing

This redbook explains the recent performance enhancements in the DFSMS/MVS Hierarchical File System as used by OS/390 UNIX System Services. It shows you how to set up and manage sharing of HFS data sets in a sysplex and gives practical examples of how to back up and recover HFS (including multi-volume HFS) data sets in a sysplex environment. The use of Tivoli Storage Manager for file-level backup is also described.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5482-01 ISBN 0738419214