

General Parallel File System



Administration and Programming Reference

Version 3 Release 2.1

General Parallel File System



Administration and Programming Reference

Version 3 Release 2.1

Note:

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 399.

Third Edition (August 2008)

| This edition applies to version 3, release 2, modification 1 of IBM General Parallel File System Multiplatform (product
| number 5724-N94), IBM General Parallel File System for POWER™ (product number 5765-G66), and to all
| subsequent releases and modifications until otherwise indicated in new editions. Technical changes or additions to
| the text and illustrations are indicated by a vertical line (|) to the left of the change.

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

International Business Machines Corporation
Department 58HA, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+845+432-9405

FAX (Other Countries): Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrcfs@us.ibm.com

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

Make sure to include the following in your comment or note:

- Title and order number of this publication
- Page number or topic related to your comment

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

© Copyright International Business Machines Corporation 1998, 2008.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Tables	ix
About this information	xi
Who should read this information	xi
Conventions used in this information	xi
Prerequisite and related information	xii
ISO 9000	xii
Using LookAt to look up message explanations	xii
How to send your comments	xiii
Summary of changes	xv
Chapter 1. Performing GPFS administration tasks	1
Requirements for administering a GPFS file system	1
Common GPFS command principles	2
Specifying nodes as input to GPFS commands	2
Chapter 2. Managing your GPFS cluster	5
Creating your GPFS cluster	5
Displaying GPFS cluster configuration information	5
Adding nodes to a GPFS cluster	6
Deleting nodes from a GPFS cluster	7
Changing the GPFS cluster configuration data	8
Node quorum considerations	12
Node quorum with tiebreaker considerations	12
Displaying and changing the file system manager node	13
Determining how long mmrestripefs takes to complete	13
Starting and stopping GPFS	14
Chapter 3. Managing file systems	17
Mounting a file system	17
Mounting a file system on multiple nodes	17
GPFS-specific mount options	17
Unmounting a file system	18
Unmounting a file system on multiple nodes	19
Deleting a file system	19
Determining which nodes have a file system mounted	19
Checking and repairing a file system	20
Listing file system attributes	21
Modifying file system attributes	22
Querying and changing file replication attributes	22
Querying file replication	23
Changing file replication attributes	23
Using Direct I/O on a file in a GPFS file system	23
Restripping a GPFS file system	24
Querying file system space	25
Querying and reducing file system fragmentation	26
Querying file system fragmentation	26
Reducing file system fragmentation	27
Backing up a file system	28
Required level of Tivoli Storage Manager	28
Setting up TSM for use by GPFS	28
Specifying TSM parameters with the mmbackup command	29

Using APIs to develop backup applications	29
Chapter 4. Managing disks	31
Displaying disks in a GPFS cluster	31
Adding disks to a file system	32
Deleting disks from a file system	33
Replacing disks in a GPFS file system.	34
Additional considerations for managing disks	35
Displaying GPFS disk states	36
Disk availability	36
Disk status	36
Changing GPFS disk states and parameters	37
Changing your NSD configuration	38
Changing NSD server usage and fallback	39
Enabling and disabling Persistent Reserve	40
Chapter 5. Managing GPFS quotas	41
Enabling and disabling GPFS quota management	41
Default quotas	42
Explicitly establishing and changing quotas	42
Checking quotas	43
Listing quotas	44
Activating quota limit checking.	45
Deactivating quota limit checking.	45
Creating file system quota reports	46
Restoring quota files	47
Chapter 6. Managing GPFS access control lists and NFS export	49
Traditional GPFS ACL administration	49
Setting traditional GPFS access control lists	50
Displaying traditional GPFS access control lists	51
Applying an existing traditional GPFS access control list	51
Changing traditional GPFS access control lists.	52
Deleting traditional GPFS access control lists	52
NFS V4 ACL administration.	52
NFS V4 ACL Syntax	53
NFS V4 ACL translation	55
Setting NFS V4 access control lists	55
Displaying NFS V4 access control lists	56
Applying an existing NFS V4 access control lists	56
Changing NFS V4 access control lists	56
Deleting NFS V4 access control lists	56
Considerations when using GPFS with NFS V4 ACLs	56
NFS and GPFS	57
Exporting a GPFS file system using NFS.	57
NFS usage of GPFS cache	59
Synchronous writing using NFS	59
Unmounting a file system after NFS export	59
NIS automount considerations.	60
Clustered NFS and GPFS on Linux	60
Chapter 7. Communicating file access patterns to GPFS.	61
Chapter 8. GPFS commands.	63
mmadddisk Command.	66
mmaddnode Command	70

mmapplypolicy Command	73
mmauth Command	78
mmbackup Command	83
mmchattr Command	86
mmchcluster Command	89
mmchconfig Command	92
mmchdisk Command	100
mmcheckquota Command	104
mmchfileset Command	107
mmchfs Command	109
mmchmgr Command	114
mmchnode Command	116
mmchnsd Command	120
mmchpolicy Command	122
mmcrcluster Command	124
mmcrfileset Command	128
mmcrfs Command	130
mmcrnsd Command	137
mmcrsnapshot Command	142
mmcrvsd Command	145
mmdefedquota Command	150
mmdefquotaoff Command	152
mmdefquotaon Command	154
mmdefragfs Command	157
mmdelacl Command	160
mmdeldisk Command	162
mmdelfileset Command	165
mmdelfs Command	168
mmdelnode Command	170
mmdelnsd Command	172
mmdelsnapshot Command	174
mmdf Command	176
mmeditACL Command	179
mmedquota Command	182
mmexportfs Command	185
mmfsck Command	187
mmfsctl Command	192
mmgetacl Command	196
mmgetstate Command	199
mmimportfs Command	202
mmlinkfileset Command	205
mmlsattr Command	207
mmlscluster Command	209
mmlsconfig Command	211
mmlsdisk Command	213
mmlsfileset Command	216
mmlsfs Command	219
mmlsmgr Command	223
mmlsmount Command	225
mmlsnsd Command	227
mmlspolicy Command	230
mmlsquota Command	232
mmlssnapshot Command	235
mmmount Command	237
mmnsddiscover Command	239
mmpmon Command	241

mmputacl Command	246
mmquotaoff Command	249
mmquotaon Command	251
mmremoteccluster Command	253
mmremotefs Command	256
mmrepquota Command	259
mmrestorefs Command	262
mmrestripefile Command	265
mmrestripefs Command	268
mmrpldisk Command	271
mmsshutdown Command	275
mmsnapdir Command	277
mmstartup Command	280
mmtracectl Command	282
mmumount Command	285
mmunlinkfileset Command	288

Chapter 9. GPFS programming interfaces	291
gpfs_acl_t Structure	293
gpfs_close_inodescan() Subroutine	294
gpfs_cmp_fssnapid() Subroutine	295
gpfs_direntx_t Structure	297
gpfs_fcntl() Subroutine	298
gpfs_fgetattrs() Subroutine	301
gpfs_fputattrs() Subroutine	303
gpfs_fputattrswithpathname() Subroutine	305
gpfs_free_fssnaphandle() Subroutine	307
gpfs_fssnap_handle_t Structure	308
gpfs_fssnap_id_t Structure	309
gpfs_fstat() Subroutine	310
gpfs_get_fsname_from_fssnaphandle() Subroutine	312
gpfs_get_fssnaphandle_by_fssnapid() Subroutine	313
gpfs_get_fssnaphandle_by_name() Subroutine	314
gpfs_get_fssnaphandle_by_path() Subroutine	316
gpfs_get_fssnapid_from_fssnaphandle() Subroutine	318
gpfs_get_pathname_from_fssnaphandle() Subroutine	320
gpfs_get_snapdirname() Subroutine	321
gpfs_get_snapname_from_fssnaphandle() Subroutine	323
gpfs_getacl() Subroutine	324
gpfs_iattr_t Structure	326
gpfs_iclose() Subroutine	329
gpfs_ifile_t Structure	330
gpfs_igetattrs() Subroutine	331
gpfs_igetfilesetname() Subroutine	333
gpfs_igetstoragepool() Subroutine	335
gpfs_iopen() Subroutine	337
gpfs_iread() Subroutine	339
gpfs_ireaddir() Subroutine	341
gpfs_ireadlink() Subroutine	343
gpfs_ireadx() Subroutine	345
gpfs_iscan_t Structure	347
gpfs_next_inode() Subroutine	348
gpfs_opaque_acl_t Structure	350
gpfs_open_inodescan() Subroutine	351
gpfs_prealloc() Subroutine	353
gpfs_putacl() Subroutine	355

gpfs_quotactl() Subroutine	357
gpfs_quotaInfo_t Structure	360
gpfs_seek_inode() Subroutine	362
gpfs_stat() Subroutine	364
gpfsAccessRange_t Structure	366
gpfsCancelHints_t Structure	367
gpfsClearFileCache_t Structure	368
gpfsDataShipMap_t Structure	369
gpfsDataShipStart_t Structure	371
gpfsDataShipStop_t Structure	374
gpfsFcntlHeader_t Structure	375
gpfsFreeRange_t Structure	376
gpfsGetFilesetName_t Structure	377
gpfsGetReplication_t Structure	378
gpfsGetSnapshotName_t Structure	380
gpfsGetStoragePool_t Structure	381
gpfsMultipleAccessRange_t Structure	382
gpfsRestripeData_t Structure	384
gpfsSetReplication_t Structure	385
gpfsSetStoragePool_t Structure	387
Chapter 10. GPFS user exits	389
mmsdrbackup User exit	390
nsdddevices User exit	391
syncfsconfig User exit	392
Chapter 11. Considerations for GPFS applications.	393
Exceptions to Open Group technical standards	393
Determining if a file system is controlled by GPFS	393
GPFS exceptions and limitations to NFS V4 ACLs	394
Chapter 12. File system format changes between versions of GPFS.	395
Accessibility features for GPFS	397
Accessibility features	397
Keyboard navigation	397
IBM and accessibility	397
Notices	399
Trademarks	400
Glossary	403
Index	407

Tables

1.	Typographic conventions	xi
2.	Configuration attributes on the mmchconfig command	9
3.	Removal of a file with ACL entries DELETE and DELETE_CHILD	54
4.	GPFS commands	63
5.	The mmeditACL command for POSIX and NFS V4 ACLs.	179
6.	The mmgetACL command for POSIX and NFS V4 ACLs	196
7.	The mmputACL command for POSIX and NFS V4 ACLs	246
8.	GPFS programming interfaces	291
9.	GPFS user exits	389

About this information

This information explains how to:

- Use the commands and programming interfaces unique to the General Parallel File System™ (GPFS™) licensed program.
- Export a GPFS file system using a the Network File System (NFS) protocol.

| This edition applies to GPFS version 3.2.1 for AIX®, Linux®, and Windows®.

To find out which version of GPFS is running on a particular AIX node, enter:

```
lslpp -l gpfs\*
```

To find out which version of GPFS is running on a particular Linux node, enter:

```
rpm -qa | grep gpfs
```

| To find out which version of GPFS is running on a particular Windows node, use the graphical user interface (GUI) and follow these steps:

- | 1. Click **Control Panel**→**Add or Remove Programs**
- | 2. Click **IBM General Parallel File System** and choose **Click here for support information**.

Throughout this information you will see various command and component names beginning with the prefix **mm**. This is not an error. GPFS shares many components with the related products IBM® Multi-Media Server and IBM Video Charger.

Who should read this information

| This information is designed for system administrators and programmers of GPFS systems. To use this
| information, you should be familiar with the GPFS licensed product and the AIX, Linux, or Windows
| operating system, or all of them, depending on which operating systems are in use at your installation.
| Where necessary, some background information relating to AIX, Linux, or Windows is provided. More
| commonly, you are referred to the appropriate documentation.

Conventions used in this information

Table 1 describes the typographic conventions used in this information.

Table 1. Typographic conventions

Typographic convention	Usage
Bold	Bold words or characters represent system elements that you must use literally, such as commands, flags, path names, directories, file names, values, and selected menu options.
<u>Bold Underlined</u>	<u>Bold Underlined</u> keywords are defaults. These take effect if you fail to specify a different keyword.
<i>Italic</i>	<ul style="list-style-type: none">• <i>Italic</i> words or characters represent variable values that you must supply.• <i>Italics</i> are also used for publication titles and for general emphasis in text.

Table 1. *Typographic conventions (continued)*

Typographic convention	Usage
Constant width	All of the following are displayed in constant width typeface: <ul style="list-style-type: none"> • Displayed information • Message text • Example text • Specified text typed by the user • Field names as displayed on the screen • Prompts from the system • References to example text
[]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
	A vertical bar separates items in a list of choices. (In other words, it means "or")
< >	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, <Enter> refers to the key on your terminal or workstation that is labeled with the word Enter.
...	An ellipsis indicates that you can repeat the preceding item one or more times.
<Ctrl-x>	The notation <Ctrl-x> indicates a control character sequence. For example, <Ctrl-c> means that you hold down the control key while pressing <c> .
\	The continuation character is used in programming examples in this information for formatting purposes.

Prerequisite and related information

For updates to this information, see publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html.

For the latest support information, see the GPFS Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. You can use LookAt from the following locations to find IBM message explanations for Clusters software products:

- The Internet. You can access IBM message explanations directly from the LookAt Web site: <http://www.ibm.com/systems/z/os/zos/bkserv/lookat/>
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

How to send your comments

Your feedback is important in helping us to produce accurate, high-quality information. If you have any comments about this information or any other GPFS documentation:

- Send your comments by e-mail to: mhvrcfs@us.ibm.com.

Include the publication title and order number, and, if applicable, the specific location of the information you have comments on (for example, a page number or a table number).

- Fill out one of the forms at the back of this publication and return it by mail, by fax, or by giving it to an IBM representative.

To contact the IBM cluster development organization, send your comments by e-mail to: cluster@us.ibm.com.

Summary of changes

The following sections summarize changes to the GPFS licensed program and the GPFS library for version 3, release 2, modification 1. Within each information unit in the library, a vertical line to the left of text and illustrations indicates technical changes or additions made to the previous edition of the book.

Summary of changes for GPFS Version 3, Release 2, Modification 1 as updated, August 2008

Changes to GPFS and to the GPFS library for version 3, release 2, modification 1 include:

- **New information**

- GPFS for Windows Multiplatform, V3.2.1 supports the Windows Server 2003 R2 operating system running on 64-bit architectures (AMD x64 / EM64T). GPFS on Windows participates in a new or existing GPFS V3.2 cluster in conjunction with AIX and Linux (32- or 64-bit) operating systems.
- Identity mapping between Windows and UNIX[®] user accounts is one of the key advancements delivered in GPFS for Windows Multiplatform. System administrators can explicitly match users and groups defined on UNIX with those defined on Windows. This allows users to maintain file ownership and access rights from either platform. System administrators are not required to define an identity map. GPFS automatically creates a mapping when one is not defined. For more information about identity mapping, see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the *General Parallel File System: Advanced Administration Guide*.
- IBM has enhanced many of the details within GPFS to support the unique semantic requirements posed by Windows. These include case insensitive names, NTFS-like file attributes, and Windows file locking. GPFS provides a bridge between a Windows and POSIX view of files, while not adversely affecting the long-standing capabilities provided on AIX and Linux operating systems.
- GPFS for Windows Multiplatform provides the same core services to parallel and serial applications as are available on AIX and Linux operating systems. GPFS allows parallel applications simultaneous access to the same files, or different files, from any node that has the GPFS file system mounted while managing a high level of control over all file system operations. System administrators and users have a consistent command interface on AIX, Linux, and Windows operating systems.

The following commands have been updated for Windows:

- **mmchfs** to add the **-t DriveLetter** option
- **mmcrfs** to add the **-t DriveLetter** option
- **mmisfs** to add the **-t** option to display the Windows drive letter
- **mmmout** to add the *DefaultDriveLetter* and *DriveLetter* parameters
- **mmumount** to add the *DefaultDriveLetter* and *DriveLetter* parameters

With few exceptions, the commands supported on the Windows operating system are identical to the commands supported on other GPFS platforms. For a list of unsupported commands, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.

- GPFS for Windows Multiplatform, V3.2.1 does not support or has restricted support for some features. For a complete list of these limitations, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.

- **Changed information:**

Minor editorial updates marked by a vertical line to the left of the text.

- **Deleted information:**

There has been no information deleted from the GPFS library for GPFS V3.2.1.

Chapter 1. Performing GPFS administration tasks

Before you can perform GPFS administration tasks, you must review topics such as getting started with GPFS, requirements for administering a GPFS file system, and common command principles.

For information on getting started with GPFS, see the *General Parallel File System: Concepts, Planning, and Installation Guide*. This includes:

1. Installing GPFS
2. GPFS cluster creation considerations
3. Configuring and tuning your system for GPFS
4. Starting GPFS
5. Network Shared Disk creation considerations
6. File system creation considerations

This guide covers the administration and maintenance of GPFS and your file systems, and includes the following topics:

1. "Requirements for administering a GPFS file system" and "Common GPFS command principles" on page 2
2. Chapter 2, "Managing your GPFS cluster," on page 5
3. Chapter 3, "Managing file systems," on page 17
4. Chapter 4, "Managing disks," on page 31
5. Chapter 5, "Managing GPFS quotas," on page 41
6. Chapter 6, "Managing GPFS access control lists and NFS export," on page 49
7. Chapter 7, "Communicating file access patterns to GPFS," on page 61
8. Chapter 8, "GPFS commands," on page 63
9. Chapter 9, "GPFS programming interfaces," on page 291
10. Chapter 10, "GPFS user exits," on page 389
11. Chapter 11, "Considerations for GPFS applications," on page 393
12. Chapter 12, "File system format changes between versions of GPFS," on page 395

For more advanced GPFS administration topics, see *General Parallel File System: Advanced Administration Guide*.

Requirements for administering a GPFS file system

Before administering your GPFS file system, ensure that your system has been properly configured.

Ensure that these requirements have been met:

- Root authority is required to perform all GPFS administration tasks except those with a function limited to listing GPFS operating characteristics or modifying individual user file attributes.
- The authentication method between nodes in the GPFS cluster must be established before any GPFS command can be issued.
 1. When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster.
 2. The default remote communication commands are **rcp** and **rsh**. If you have designated the use of a different remote communication program, you must ensure:
 - a. Proper authorization is granted to all nodes in the GPFS cluster.
 - b. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

| **Note:** The **rsh** and **rcp** commands provided by the Windows SUA environment do not support GPFS.
| Therefore, if your cluster will include Windows nodes, you have to designate alternate remote
| programs (for example, **ssh** and **scp**).

| Common GPFS command principles

| There are some common principles that you should keep in mind when you are running GPFS commands.

| Those principles include:

- | • Unless otherwise noted, GPFS commands can be run from any node in the cluster. Exceptions are commands that are not supported in a particular operating system environment. Certain commands may additionally require the affected file system to be mounted.

| **Note:** In clusters that contain Windows nodes, it is recommended that you perform the system administration tasks from one of the AIX or Linux nodes.

- | • GPFS supports the "no" prefix on all Boolean type long (or dash-dash) options.

| Specifying nodes as input to GPFS commands

| Many GPFS commands accept a node or multiple nodes as part of their input, using the **-N** flag. Nodes can be specified to GPFS commands in a variety of ways:

| *Node* A representation of an individual node, which can be any of these:

- | • Short GPFS administration node interface name.
- | • Long GPFS administration node interface name.
- | • Short GPFS daemon node interface name.
- | • Long GPFS daemon node interface name.
- | • IP address corresponding to the GPFS daemon node interface.
- | • GPFS node number.

| *Node - Node*

| A node range, indicated by specifying two node numbers separated by a hyphen (-), with the first node number being less than or equal to the second node number. For example, node range **3-8** specifies the nodes with node numbers 3, 4, 5, 6, 7, and 8.

| *NodeClass*

| A set of nodes that share a particular characteristic. These node classes are known to GPFS:

| **all** All of the nodes in the GPFS cluster.

| **clientnodes**

| All nodes that do not participate in file system administration activities.

| **managernodes**

| All nodes in the pool of nodes from which cluster managers, file system managers, and token managers are selected.

| **mount**

| For commands involving a file system, all of the nodes in the GPFS cluster on which the file system is mounted.

| **nonquorumnodes**

| All of the non-quorum nodes in the GPFS cluster.

| **nsdnodes**

| All of the NSD server nodes in the GPFS cluster.

| **quorumnodes**

| All of the quorum nodes in the GPFS cluster.

| *NodeFile*

| A file that contains a list of nodes. A node file can contain individual nodes or node ranges.

| Not every GPFS command supports all of the above node specification options. To learn what kinds of
| node specifications are supported by a particular GPFS command, see the relevant command description
| in Chapter 8, “GPFS commands,” on page 63.

Chapter 2. Managing your GPFS cluster

There are several tasks involved in managing your GPFS cluster. This topic points you to the information you need to get started.

GPFS cluster management tasks include:

- “Creating your GPFS cluster”
- “Displaying GPFS cluster configuration information”
- “Specifying nodes as input to GPFS commands” on page 2
- “Adding nodes to a GPFS cluster” on page 6
- “Deleting nodes from a GPFS cluster” on page 7
- “Changing the GPFS cluster configuration data” on page 8
- “Node quorum considerations” on page 12
- “Node quorum with tiebreaker considerations” on page 12
- “Displaying and changing the file system manager node” on page 13
- “Determining how long mmrestripefs takes to complete” on page 13
- “Starting and stopping GPFS” on page 14

Creating your GPFS cluster

You must first create a GPFS cluster by issuing the **mmcrcluster** command.

See the *General Parallel File System: Concepts, Planning, and Installation Guide* for details on how GPFS clusters are created and used.

Displaying GPFS cluster configuration information

When managing your GPFS cluster, you can display the current configuration information for the cluster by issuing the **mmfsccluster** command.

The command displays:

- The cluster name
- The cluster id
- GPFS UID domain
- The remote shell command being used
- The remote file copy command being used
- The primary GPFS cluster configuration server
- The secondary GPFS cluster configuration server
- A list of nodes belonging the GPFS cluster

For each node, the command displays:

- The node number assigned to the node by GPFS
- Daemon node name
- Network IP address
- Admin node name
- Designation, such as whether the node is a quorum node, a manager node, or both

To display this information, enter:

```
mmfsccluster
```

The system displays information similar to:

```
GPFS cluster information
=====
```

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

```
GPFS cluster configuration servers:
-----
```

```
Primary server:  k164sn06.kgn.ibm.com
Secondary server: k164n04.kgn.ibm.com
```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.69	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164sn06.kgn.ibm.com	quorum-manager

See the “mmlscluster Command” on page 209 for complete usage information.

Adding nodes to a GPFS cluster

You can add nodes to an existing GPFS cluster by issuing the **mmaddnode** command. The new nodes are available immediately after the successful completion of this command.

You must follow these rules when adding nodes to a GPFS cluster:

- You may issue the command only from a node that already belongs to the GPFS cluster.
- A node may belong to only one GPFS cluster at a time.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command to add those nodes.

To add node **k164n06** to the GPFS cluster, enter:

```
mmaddnode -N k164n06
```

The system displays information similar to:

```
Mon Aug  9 21:53:30 EDT 2004: 6027-1664 mmaddnode: Processing node k164n06.kgn.ibm.com
mmaddnode: Command successfully completed
mmaddnode: 6027-1371 Propagating the changes to all affected nodes.
This is an asynchronous process.
```

To confirm the addition of the nodes, enter:

```
mmlscluster
```

The system displays information similar to:

```
GPFS cluster information
=====
```

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

```
GPFS cluster configuration servers:
-----
```

```
Primary server:  k164n04.kgn.ibm.com
Secondary server: k164n05.kgn.ibm.com
```


Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.69	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164n06.kgn.ibm.com	

See the “**mmaddnode** Command” on page 70 and the “**mmlscluster** Command” on page 209 for complete usage information.

Deleting nodes from a GPFS cluster

You can delete nodes from a GPFS cluster by issuing the **mmdeinode** command.

You must follow these rules when deleting nodes:

- A node being deleted cannot be the primary or secondary GPFS cluster configuration server unless you intend to delete the entire cluster. Verify this by issuing the **mmlscluster** command. If a node to be deleted is one of the servers and you intend to keep the cluster, issue the **mmchcluster** command to assign another node as a configuration server before deleting the node.
- A node that is being deleted cannot be designated as an NSD server for any disk in the GPFS cluster, unless you intend to delete the entire cluster. Verify this by issuing the **mmlsnsd** command. If a node that is to be deleted is an NSD server for one or more disks, move the disks to nodes that will remain in the cluster. Use the **mmchnsd** command to assign new NSD servers for those disks.
- GPFS must be shut down on the nodes being deleted. Use the **mmshutdown** command.

To delete the nodes listed in a file called **nodes_to_delete**, enter:

```
mmdeinode -N /tmp/nodes_to_delete
```

where **nodes_to_delete** contains the nodes **k164n01** and **k164n02**. The system displays information similar to:

```
Verifying GPFS is stopped on all affected nodes ...
mmdeinode: Command successfully completed
mmdeinode: 6027-1371 Propagating the cluster configuration data to
all affected nodes. This is an asynchronous process.
```

To confirm the deletion of the nodes, enter:

```
mmlscluster
```

The system displays information similar to:

```
GPFS cluster information
=====
```

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

```
GPFS cluster configuration servers:
```

```
-----
Primary server:  k164sn06.kgn.ibm.com
Secondary server: k164n05.kgn.ibm.com
```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.69	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164sn06.kgn.ibm.com	quorum-manager

See the “**mmdeinode** Command” on page 170 and the “**mmlscluster** Command” on page 209 for complete usage information.

Exercise caution when shutting down GPFS on quorum nodes or deleting quorum nodes from the GPFS cluster. If the number of remaining quorum nodes falls below the requirement for a quorum, you will be unable to perform file system operations. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *quorum*.

Changing the GPFS cluster configuration data

Once your GPFS cluster has been configured (see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the **mmcrcluster** command), you can change configuration attributes by issuing the **mmchcluster** or **mmchconfig** commands.

Use the **mmchcluster** command to:

- Change the name of the cluster.
- Change the primary or secondary GPFS cluster configuration server nodes. The primary or secondary server may be changed to another node in the GPFS cluster. That node must be available for the command to be successful.

Attention: If during the change to a new primary or secondary GPFS cluster configuration server, one or both of the old server nodes are down, it is imperative that you run the **mmchcluster -p LATEST** command as soon as the old servers are brought back online. Failure to do so may lead to disruption in GPFS operations.

- Synchronize the primary GPFS cluster configuration server node. If an invocation of the **mmchcluster** command fails, you will be prompted to reissue the command and specify **LATEST** on the **-p** option to synchronize all of the nodes in the GPFS cluster. Synchronization instructs all nodes in the GPFS cluster to use the most recently specified primary GPFS cluster configuration server.
- Change the remote shell and remote file copy programs to be used by the nodes in the cluster. These commands must adhere to the syntax forms of the **rsh** and **rcp** commands, but may implement an alternate authentication mechanism.

For example, to change the primary server for the GPFS cluster data, enter:

```
mmchcluster -p k164n06
```

The system displays information similar to:

```
mmchcluster -p k164n06
mmchcluster: Command successfully completed
```

To confirm the change, enter:

```
mmfsccluster
```

The system displays information similar to:

GPFS cluster information

=====

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

GPFS cluster configuration servers:

```
Primary server:   k164sn06.kgn.ibm.com
Secondary server: k164n05.kgn.ibm.com
```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.69	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164sn06.kgn.ibm.com	quorum-manager

Attention: The **mmchcluster** command, when issued with either the **-p** or **-s** option, is designed to operate in an environment where the current primary and secondary GPFS cluster configuration servers are *not* available. As a result, the command can run without obtaining its regular serialization locks. To assure smooth transition to a new cluster configuration server, no other GPFS commands (**mm...** commands) should be running when the command is issued nor should any other command be issued until the **mmchcluster** command has successfully completed.

See the “mmchcluster Command” on page 89 and the “mmlscluster Command” on page 209 for complete usage information.

You may be able to tune your cluster for better performance by re-configuring one or more attributes. Before you change any attributes, consider how the changes will affect the operation of GPFS. For a detailed discussion see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the **mmcrcluster** command.

Table 2 details the GPFS cluster configuration attributes which can be changed by issuing the **mmchconfig** command. Variations under which these changes take effect are noted:

1. Take effect immediately and are permanent (**-i**).
2. Take effect immediately but do not persist when GPFS is restarted (**-l**).
3. Require that the GPFS daemon be stopped on all nodes for the change to take effect.
4. May be applied to only a subset of the nodes in the cluster.

Table 2. Configuration attributes on the **mmchconfig** command

Attribute name and Description	-i option allowed	-l option allowed	GPFS must be stopped on all nodes	List of NodeNames allowed	Change takes effect
autoload Starting GPFS automatically	no	no	no	no	on reboot of each node
automountDir Name of the automount directory	no	no	yes	no	on restart of the daemon
cipherList When set, GPFS security using OpenSSL is enabled.	no	no	yes	no	on restart of the daemon
cnfsMountdPort The port number to be used for rpc.mountd	yes	yes	no	no	if not immediately, on restart of the daemon
cnfsNFSDprocs The number of nfds kernel threads	yes	yes	no	no	if not immediately, on restart of the daemon
cnfsSharedRoot Directory to be used by the clustered NFS subsystem	no	no	no	yes	on restart of the daemon
cnfsVIP Virtual DNS name for the list of CNFS IP addresses	yes	yes	no	no	if not immediately, on restart of the daemon
dataStructureDump Path for the storage of dumps	yes	yes	no	yes	if not immediately, on restart of the daemon

Table 2. Configuration attributes on the **mmchconfig** command (continued)

Attribute name and Description	-i option allowed	-l option allowed	GPFS must be stopped on all nodes	List of NodeNames allowed	Change takes effect
defaultMountDir Default parent directory for GPFS file systems	yes	yes	no	no	for new file systems
dmapiEventTimeout DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiMountEvent DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiMountTimeout DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
dmapiSessionFailureTimeout DMAPI attribute	yes	yes	no	yes	if not immediately, on restart of the daemon
failureDetectionTime Indicates the amount of time it will take to detect that a node has failed (when persistent reserve (PR) is enabled)	no	no	no	no	on restart of the daemon
When using this option, the GPFS daemon must be shut down on all nodes.					
maxblocksize Maximum file system block size allowed	no	no	no	yes	on restart of the daemon
maxFilesToCache Number of inodes to cache for recently used files	no	no	no	yes	on restart of the daemon
maxMBpS I/O throughput estimate	yes	yes	no	yes	if not immediately, on restart of the daemon
maxStatCache Number of inodes to keep in stat cache	no	no	no	yes	on restart of the daemon
nsdServerWaitTimeForMount Number of seconds to wait for an NSD server to come up	yes	yes	no	yes	if not immediately, on restart of the daemon
nsdServerWaitTimeWindowOnMount Time window to determine if quorum is to be considered <i>recently formed</i>	yes	yes	no	yes	if not immediately, on restart of the daemon
pagepool Size of buffer cache on each node	yes	yes	no	yes	if not immediately, on restart of the daemon

Table 2. Configuration attributes on the **mmchconfig** command (continued)

Attribute name and Description	-i option allowed	-l option allowed	GPFS must be stopped on all nodes	List of NodeNames allowed	Change takes effect
pagepoolMaxPhysMemPct Percentage of physical memory that can be assigned to the page pool	no	no	no	yes	on restart of the daemon
prefetchThreads Maximum number of threads dedicated to prefetching data	no	no	no	yes	on restart of the daemon
release=LATEST Complete the migration to a new release	yes	no	no	no	if not immediately, on restart of the daemon
sidAutoMapRangeLength Controls the length of the reserved range for Windows SID to UNIX ID mapping	yes	yes	no	no	if not immediately, on restart of the daemon
sidAutoMapRangeStart Specifies the start of the reserved range for Windows SID to UNIX ID mapping	no	no	no	no	on restart of the daemon
subnets List of subnets to be used for most efficient daemon-to-daemon communication	no	no	no	no	on restart of the daemon
tiebreakerDisks List of tie breaker disks (NSDs)	no	no	yes	no	on restart of the daemon
uidDomain The UID domain name for the cluster.	no	no	yes	no	on restart of the daemon
unmountOnDiskFail Unmount the file system on a disk failure	yes	yes	no	yes	if not immediately, on restart of the daemon
usePersistentReserve Enables or disables persistent reserve (PR) on the disks	no	no	yes	no	on restart of the daemon
verbsPorts Specifies InfiniBand device names and port numbers	no	no	no	yes	on restart of the daemon
verbsRdma Enables or disables InfiniBand RDMA using the Verbs API	yes	yes	no	yes	if not immediately, on restart of the daemon

Table 2. Configuration attributes on the **mmchconfig** command (continued)

Attribute name and Description	-i option allowed	-l option allowed	GPFS must be stopped on all nodes	List of NodeNames allowed	Change takes effect
worker1Threads maximum number of concurrent file operations	no	no	no	yes	on restart of the daemon

Specify the nodes you want to target for change and the attributes with their new values on the **mmchconfig** command. For example, to change the **pagepool** value for each node in the GPFS cluster immediately, enter:

```
mmchconfig pagepool=100M -i
```

The system displays information similar to:

```
mmchconfig: Command successfully completed
mmchconfig: 6027-1371 Propagating the changes to all affected nodes.
This is an asynchronous process.
```

See the “mmchconfig Command” on page 92 for complete usage information.

Node quorum considerations

A node quorum is the minimum number of nodes that must be running in order for the daemon to start. Node quorum is the default quorum algorithm for GPFS.

For a discussion on node quorum, see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *node quorum*.

Node quorum with tiebreaker considerations

For a discussion on node quorum with tiebreaker, see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *node quorum with tiebreaker*. In addition, review the information in this topic.

When using node quorum with tiebreaker, define between one, two, or three disks to be used as tiebreaker disks when any quorum node is down. Issue this command:

```
mmchconfig tiebreakerDisks="nsdName;nsdName;nsdName"
```

Consider these points:

- You are not permitted to change a GPFS cluster configuration to use node quorum with tiebreaker if there are more than eight existing quorum nodes.
- You can have a maximum of three tiebreaker disks.
- The disks must have one of these types of attachments to the quorum nodes:
 - Fibre-channel SAN
 - Virtual shared disks
- The GPFS daemons must be down on all nodes in the cluster when running **mmchconfig tiebreakerDisks**.

If you are using node quorum with tiebreaker and want to change to using node quorum, issue this command:

```
mmchconfig tiebreakerDisks=DEFAULT
```

Displaying and changing the file system manager node

In general, GPFS performs the same functions on all nodes. There are also cases where one node provides a more global function that affects the operation of multiple nodes. For example, each file system is assigned a node that functions as a file system manager.

For a more detailed discussion on the role of the file system manager node, see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *Special management functions*.

The node that is the file system manager can also be used for applications. In some cases involving very large clusters or applications that place a high stress on metadata operations, it may be useful to specify which nodes are used as file system managers. Applications that place a high stress on metadata operations are usually those that involve large numbers of very small files, or that do very fine-grain parallel write-sharing among multiple nodes.

You can display the file system manager node by issuing the **mmlsmgr** command. You can display the information for an individual file system, a list of file systems, or for all of the file systems in the cluster. For example, to display the file system manager for the file system **fs1**, enter:

```
mmlsmgr fs1
```

The output shows the device name of the file system and the file system manager's node number and name:

```
file system      manager node      [from 19.134.68.69 (k164n05)]
-----
fs1              19.134.68.70 (k164n06)
```

See the “mmlsmgr Command” on page 223 for complete usage information.

You can change the file system manager node for an individual file system by issuing the **mmchmgr** command. For example, to change the file system manager node for the file system **fs1** to **k145n32**, enter:

```
mmchmgr fs1 k145n32
```

The output shows the file system manager's node number and name, in parentheses, as recorded in the GPFS cluster data:

```
GPFS: 6027-628 Sending migrate request to current manager node 19.134.68.69 (k145n30).
GPFS: 6027-629 Node 19.134.68.69 (k145n30) resigned as manager for fs1.
GPFS: 6027-630 Node 19.134.68.70 (k145n32) appointed as manager for fs1.
```

See the “mmchmgr Command” on page 114 for complete usage information.

Determining how long mmrestripefs takes to complete

There are several factors that determine how long the **mmrestripefs** command takes to complete.

To determine how long the **mmrestripefs** command will take to complete, consider these points:

1. How much data is to be moved by issuing the **df -k** command.
2. How many GPFS client nodes there are to do the work.
3. How much virtual shared disk server or Network Shared Disk (NSD) server bandwidth is available for I/O.
4. If you have added new disks to a file system, after the disks have been added, determine how much free space is on each of the new disks by issuing the **mmdf Device** command.

The restriping of a file system is done by having one thread on each node in the cluster work on a subset of files. Consequently, the more GPFS client nodes there are performing work for the restripe, the faster the **mmrestripefs** command will complete. The nodes that should participate in the restripe are specified

on the command using the **-N** parameter. Based on raw I/O rates, you should be able to estimate the length of time for the restripe. However, to account for the overhead of scanning all metadata, that value should be doubled.

Assuming that you have enough nodes to saturate the disk servers, and have to move all of the data, the time to read and write every block of data is roughly:

$$2 * \text{fileSystemSize} / \text{averageDiskserverDataRate}$$

As an upper bound, due to overhead of scanning all of the metadata, this time should be doubled. If other jobs are loading the virtual shared disk servers heavily, this time may increase even more.

Note: There is no particular reason to stop all other jobs while the **mmrestripefs** command is running. The CPU load of the command is minimal on each node and only the files that are being restriped at any moment are locked to maintain data integrity.

Starting and stopping GPFS

Use the **mmstartup** and **mmshutdown** commands to start and stop GPFS on new or existing clusters. For new GPFS clusters, see *Steps to establishing and starting your GPFS cluster* in: *General Parallel File System: Concepts, Planning, and Installation Guide*.

For existing GPFS clusters, before starting GPFS, ensure that you have:

1. Verified the installation of all prerequisite software.
2. Compiled the GPL layer, if Linux is being used.
3. Properly configured and tuned your system for use by GPFS. This should be done prior to starting GPFS.

For details, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.

Start the daemons on all of the nodes in the cluster by issuing the **mmstartup -a** command:

```
mmstartup -a
```

The output is similar to this:

```
Tue Aug 24 15:54:56 edt 2004: 6027-1642 mmstartup: Starting GPFS ...
```

Check the messages recorded in **/var/adm/ras/mmfs.log.latest** on one node for verification. Look for messages similar to this:

```
mmfsd initializing ...  
GPFS: 6027-300 mmfsd ready
```

This indicates that quorum has been formed and this node has successfully joined the cluster, and is now ready to mount file systems.

If GPFS does not start, see the *GPFS: Problem Determination Guide* and search on *GPFS daemon will not come up*.

See the “mmstartup Command” on page 280 for complete usage information.

If it becomes necessary to stop GPFS, you can do so from the command line by issuing the **mmshutdown** command:

```
mmshutdown -a
```

The system displays information similar to:


```
Thu Aug 12 13:10:40 EDT 2004: 6027-1341 mmshutdown: Starting force unmount of GPFS file systems
k164n05.kgn.ibm.com: forced unmount of /fs1
k164n04.kgn.ibm.com: forced unmount of /fs1
k164n06.kgn.ibm.com: forced unmount of /fs1
Thu Aug 12 13:10:45 EDT 2004: 6027-1344 mmshutdown: Shutting down GPFS daemons
k164n04.kgn.ibm.com: Shutting down!
k164n06.kgn.ibm.com: Shutting down!
k164n05.kgn.ibm.com: Shutting down!
k164n04.kgn.ibm.com: 'shutdown' command about to kill process 49682
k164n05.kgn.ibm.com: 'shutdown' command about to kill process 28194
k164n06.kgn.ibm.com: 'shutdown' command about to kill process 30782
Thu Aug 12 13:10:54 EDT 2004: 6027-1345 mmshutdown: Finished
```

See the “mmshutdown Command” on page 275 for complete usage information.

Chapter 3. Managing file systems

There are several file system management tasks outlined in this topic. If you need information about how to create GPFS file systems, see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the **mmcrfs** command.

File system management tasks include:

1. "Mounting a file system"
2. "Unmounting a file system" on page 18
3. "Deleting a file system" on page 19
4. "Determining which nodes have a file system mounted" on page 19
5. "Checking and repairing a file system" on page 20
6. "Listing file system attributes" on page 21
7. "Modifying file system attributes" on page 22
8. "Querying and changing file replication attributes" on page 22
9. "Restriping a GPFS file system" on page 24
10. "Querying file system space" on page 25
11. "Querying and reducing file system fragmentation" on page 26
12. "Backing up a file system" on page 28

Managing filesets, storage pools and policies is also a file system management task. See *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide* for more information.

Mounting a file system

You must explicitly mount a GPFS file system if this is the first time the file system is being mounted after its creation, or you specified *not to* automatically mount (**-A no**) the file system when you created it.

If you allowed the default value for the automatic mount option (**-A yes**) when you created the file system, then you do not need to use this procedure after restarting GPFS on the nodes.

To mount a GPFS file system, enter:

```
mmmount device
```

where *device* is the name of the file system. For example, to mount the file system **fs1**, enter:

```
mmmount fs1
```

Mounting a file system on multiple nodes

To mount file system **fs1** on all nodes in the GPFS cluster, issue this command:

```
mmmount fs1 -a
```

To mount a file system only on a specific set of nodes, use the **-N** flag of the **mmmount** command.

GPFS-specific mount options

GPFS-specific mount options can be specified with the **-o** parameter on the **mmchfs**, **mmremotefs**, **mmmount** and **mount** commands. Options specified with the **mmchfs** and **mmremotefs** commands are

recorded in the GPFS configuration files and are passed as default options to subsequent mount commands on all nodes in the cluster. Options specified with the **mmmount** or **mount** commands override the existing default settings, and are not persistent.

All of the mount options can be specified using the **-o** parameter. Multiple options should be separated only by a comma. If an option is specified multiple times, the last instance is the one that takes effect. Certain options can also be set with specifically designated command flags. Unless otherwise stated, mount options can be specified as:

option or *option=1* or *option=yes* - to enable the option

nooption or *option=0* or *option=no* - to disable the option

The *option={1 | 0 | yes | no}* syntax should be used for options that can be intercepted by the **mount** command and not passed through to GPFS. An example is the **atime** option in the Linux environment.

The GPFS-specific mount options are:

atime Update inode access time for each access. This is the default. This option can also be controlled with the **-S** option on the **mmcrfs** and **mmchfs** commands.

mtime Always return accurate file modification times. This is the default. This option can also be controlled with the **-E** option on the **mmcrfs** and **mmchfs** commands.

noatime

Do not update inode access times on this file system. This option can also be controlled with the **-S** option on the **mmcrfs** and **mmchfs** commands.

nomtime

Update file modification times only periodically. This option can also be controlled with the **-E** option on the **mmcrfs** and **mmchfs** commands.

nosyncnfs

Do not commit metadata changes coming from the NFS daemon synchronously. Normal file system synchronization semantics apply. This is the default.

syncnfs

Synchronously commit metadata changes coming from the NFS daemon.

useNSDserver={always | asfound | asneeded | never}

Controls the initial disk discovery and failover semantics for NSD disks. The possible values are:

always

Always access the disk using the NSD server. Local dynamic disk discovery is disabled.

asfound

Access the disk as found (the first time the disk was accessed). No change of disk access from local to NSD server, or the other way around, is performed by GPFS.

asneeded

Access the disk any way possible. This is the default.

never Always use local disk access.

Unmounting a file system

Some GPFS administration tasks require you to unmount the file system before they can be performed. You can unmount a GPFS file system using the **mmumount** command.

If the file system will not unmount, see the *General Parallel File System: Problem Determination Guide* and search for *file system will not unmount*.

To unmount a GPFS file system using the **mmumount** command, enter:

```
mmumount device
```

where *device* is the name of the file system. For example, to unmount the file system **fs1**, enter:

```
mmumount fs1
```

Unmounting a file system on multiple nodes

To unmount file system **fs1** on all nodes in the GPFS cluster, issue this command:

```
mmumount fs1 -a
```

To unmount a file system only on a specific set of nodes, use the **-N** flag of the **mmumount** command.

Deleting a file system

Before deleting a file system, unmount it on all nodes.

Specify the file system to be deleted on the **mmdelfs** command. For example, to delete the file system **fs1**, enter:

```
mmdelfs fs1
```

The system displays information similar to:

```
mmdelfs: 6027-1366 Marking the disks as available
GPFS: 6027-573 All data on following disks of fs1 will be destroyed:
    gpfs9nsd
    gpfs13nsd
    gpfs11nsd
    gpfs12nsd
GPFS: 6027-574 Completed deletion of file system fs1.
mmdelfs: 6027-1371 Propagating the changes to all affected nodes.
This is an asynchronous process.
```

For more information, see the following:

- “Unmounting a file system” on page 18
- “mmdelfs Command” on page 168 for complete usage information
- “mmdelnsd Command” on page 172 for removing the NSD definitions after deleting the file system

Determining which nodes have a file system mounted

The **mmlsmount** command is used to determine which nodes have a given file system mounted. The name and IP address of each node that has the file system mounted is displayed. This command can be used for all file systems, all remotely mounted file systems, or file systems mounted on nodes of certain clusters.

Note that the **mmlsmount -L** command reports file systems that are in use at the time the command is issued. A file system is considered to be in use if it is explicitly mounted with the **mount** or **mmmount** command or if it is mounted internally for the purposes of running some other GPFS command. For example, when you run the **mmrestripefs** command, the file system will be internally mounted for the duration of the command. If **mmlsmount** is issued in the interim, the file system will be reported as being in use by the **mmlsmount** command but, unless it is explicitly mounted, will not show up in the output of the **mount** or **df** commands.

This is an example of a **mmlsmount -L** command for a locally mounted file system named **fs1**:

```
File system fs1 (mnsd.cluster:fs1) is mounted on 5 nodes:
9.114.132.101   c5n101           mnsd.cluster
9.114.132.100   c5n100           mnsd.cluster
9.114.132.106   c5n106           mnsd.cluster
9.114.132.97    c5n97            cluster1.cluster
9.114.132.92    c5n92            cluster1.cluster
```

Checking and repairing a file system

The **mmfsck** command finds and repairs conditions that can cause problems in your file system. The **mmfsck** command operates in two modes: online and offline.

The online mode operates on a mounted file system and is chosen by issuing the **-o** option. Conversely, the offline mode operates on an unmounted file system. In general it is unnecessary to run **mmfsck** in offline mode unless under the direction of the IBM Support Center.

The online mode checks and recovers unallocated blocks on a mounted file system. If a GPFS file operation fails due to an out of space condition, the cause may be disk blocks that have become unavailable after repeated node failures. The corrective action taken is to mark the block free in the allocation map. Any other inconsistencies found are only reported, not repaired.

Note:

1. If you are running the online **mmfsck** command to free allocated blocks that do not belong to any files, plan to make file system repairs when system demand is low. This is I/O intensive activity and it can affect system performance.
2. If you are repairing a file system due to node failure and the file system has quotas enabled, it is suggested that you run the **mmcheckquota** command to recreate the quota files.

To repair any other inconsistencies, you must run the offline mode of the **mmfsck** command on an unmounted file system. The offline mode checks for these file inconsistencies that might cause problems:

- Blocks marked allocated that do not belong to any file. The corrective action is to mark the block free in the allocation map.
- Files and directories for which an inode is allocated and no directory entry exists, known as orphaned files. The corrective action is to create directory entries for these files in a **lost+found** subdirectory in the root directory of the fileset to which the file or directory belongs. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. The index number of the inode is assigned as the name. If you do not allow the **mmfsck** command to reattach an orphaned file, it asks for permission to delete the file.
- Directory entries pointing to an inode that is not allocated. The corrective action is to remove the directory entry.
- Incorrectly formed directory entries. A directory file contains the inode number and the generation number of the file to which it refers. When the generation number in the directory does not match the generation number stored in the file's inode, the corrective action is to remove the directory entry.
- Incorrect link counts on files and directories. The corrective action is to update them with accurate counts.
- Policy files that are not valid. The corrective action is to delete the file.
- Various problems related to filesets: missing or corrupted fileset metadata, inconsistencies in directory structure related to filesets, missing or corrupted fileset root directory, other problems in internal data structures.

The **mmfsck** command performs other functions not listed here, as deemed necessary by GPFS.

You cannot run the **mmfsck** command on a file system that has disks in a **down** state. You must first run the **mmchdisk** command to change the state of the disks to **unrecovered** or **up**. To display the status of the disks in the file system, issue the **mmfsdisk** command.

For example, to check the file system **fs1** without making any changes to the file system, enter:

```
mmfsck fs1
```

The system displays information similar to:

```
Checking "fs1"
Checking inodes
Checking inode map file
Checking directories and files
Checking log files
Checking extended attributes file
Checking allocation summary file
Checking policy file
Validated policy 'for stripe group fs1': parsed 3 Placement Rules,
0 Migrate/Delete/Exclude Rules
Migrate/Delete/Exclude Rules
Checking filesets metadata
Checking file reference counts
Checking file system replication status
```

```
1212416 inodes
  87560 allocated
    0 repairable
    0 repaired
    0 damaged
    0 deallocated
    0 orphaned
    0 attached
```

```
7211746 subblocks
  227650 allocated
    0 unreferenced
    0 deletable
    0 deallocated
```

```
144504 addresses
    0 suspended
```

File system is clean.

See the “mmchdisk Command” on page 100, “mmcheckquota Command” on page 104, “mmfsck Command” on page 187, and “mmlsdisk Command” on page 213 for complete usage information.

Listing file system attributes

Use the **mmlsfs** command to display the current file system attributes. Depending on your configuration, additional information which is set by GPFS may be displayed to assist in problem determination when contacting the IBM Support Center.

If you specify no options with the **mmlsfs** command, all file system attributes are listed.

For example, to list all of the attributes for the file system **fs2**, enter:

```
mmlsfs fs2
```

The system displays information similar to:

flag	value	description
-f	512	Minimum fragment size in bytes
-i	512	Inode size in bytes
-I	8192	Indirect block size in bytes
-m	2	Default number of metadata replicas
-M	2	Maximum number of metadata replicas
-r	1	Default number of data replicas

```

| -R 2           Maximum number of data replicas
| -j cluster     Block allocation type
| -D posix       File locking semantics in effect
| -k all         ACL semantics in effect
| -a -1         Estimated average file size
| -n 1000        Estimated number of nodes that will mount file system
| -B 16384       Block size
| -Q user;group;fileset Quotas enforced
|               none      Default quotas enabled
| -F 3000096     Maximum number of inodes
| -V 10.00(3.2.0.0) File system version. Highest supported version: 10.00
| -u yes         Support for large LUNs?
| -z no          Is DMAPI enabled?
| -L 524288      Logfile size
| -E yes         Exact mtime mount option
| -S no          Suppress atime mount option
| -K whenpossible Strict replica allocation option
| -P system      Disk storage pools in file system
| -d hd3n97;hd4n97;hd5n98;hd6n98;hd7vsgn97;hd8vsgn97 Disks in file system
| -A no          Automatic mount option
| -o none        Additional mount options
| -T /fs2        Default mount point
| -t U           Default Windows drive letter

```

See the “mmlsfs Command” on page 219 for complete usage information. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *GPFS architecture* and *file system creation considerations* for a detailed discussion of file system attributes.

Modifying file system attributes

Use the **mmchfs** command to modify existing file system attributes.

Note: All files created after issuing the **mmchfs** command take on the new attributes. Existing files are not affected. Use the **mmchattr** or **mmrestripefs -R** command to change the replication factor of existing files. See “Querying and changing file replication attributes.”

For example, to change the default data replication factor to 2 for the file system **fs1**, enter:

```
mmchfs fs1 -r 2
```

To confirm the changes, enter:

```
mmlsfs fs1 -r
```

The system displays information similar to:

flag	value	description
-r	2	Default number of data replicas

See the “mmchfs Command” on page 109 and the “mmlsfs Command” on page 219 for complete usage information. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *GPFS architecture* and *file system creation considerations* for a detailed discussion of file system attributes.

Querying and changing file replication attributes

If your availability requirements change, you can have GPFS display the current replication factors for one or more files by issuing the **mmlsattr** command. You might then decide to change replication for one or more files using the **mmchattr** command.

For complete usage information, see the “mmlsattr Command” on page 207 and the “mmchattr Command” on page 86.

Querying file replication

Specify one or more file names with the **mmlsattr** command. For example, to display the replication factors for two files named **project4.sched** and **project4.resource** in the file system **fs1**, enter:

```
mmlsattr /fs1/project4.sched /fs1/project4.resource
```

The system displays information similar to:

```
replication factors
metadata(max) data(max) file      [flags]
-----
1 ( 2) 1 ( 2) /fs1/project4.sched
1 ( 2) 1 ( 2) /fs1/project4.resource
```

See the “mmlsattr Command” on page 207 for complete usage information. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *GPFS architecture* and *file system creation considerations* for a detailed discussion of file system attributes.

Changing file replication attributes

Use the **mmchattr** command to change the replication attributes for one or more files.

You can only increase data and metadata replication as high as the maximum data and maximum metadata replication factors for that file system. You cannot change the maximum data and maximum metadata replication factors once the file system has been created.

Specify the file name, attribute, and new value with the **mmchattr** command. For example, to change the metadata replication factor to 2 and the data replication factor to 2 for the file named **project7.resource** in the file system **fs1**, enter:

```
mmchattr -m 2 -r 2 /fs1/project7.resource
```

To confirm the change, enter:

```
mmlsattr /fs1/project7.resource
```

The system displays information similar to:

```
replication factors
metadata(max) data(max) file      [flags]
-----
2 ( 2) 2 ( 2) /fs1/project7.resource
```

See the “mmchattr Command” on page 86 and the “mmlsattr Command” on page 207 for complete usage information. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *GPFS architecture* and *file system creation considerations* for a detailed discussion of file system attributes.

Using Direct I/O on a file in a GPFS file system

The Direct I/O caching policy can be set for files in a GPFS file system by specifying the **-D** option on the **mmchattr** command.

This caching policy bypasses file cache and transfers data directly from disk into the user space buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Applications with poor cache hit rates or very large I/Os may benefit from the use of Direct I/O.

Direct I/O may also be specified by supplying the **O_DIRECT** file access mode on the **open()** of the file.

Restripping a GPFS file system

Writing data into a GPFS file system correctly stripes the file. However, if you have added disks to a GPFS file system that are seldom updated, use the **mmrestripefs** command to restripe the file system to achieve maximum performance.

Restripping offers the opportunity to specify useful options in addition to rebalancing (**-b** option). Re-replicating (**-r** or **-R** option) provides for proper replication of all data and metadata. If you use replication, this option is useful to protect against additional failures after losing a disk. For example, if you use a replication factor of 2 and one of your disks fails, only a single copy of the data would remain. If another disk then failed before the first failed disk was replaced, some data might be lost. If you expect delays in replacing the failed disk, you could protect against data loss by suspending the failed disk using the **mmchdisk** command and re-replicating. This would assure that all data existed in two copies on operational disks.

If files are assigned to one storage pool, but with data in a different pool, the placement (**-p**) option will migrate their data to the correct pool. Such files are referred to as ill-placed. Utilities, such as the **mmchattr** command, may change a file's storage pool assignment, but not move the data. The **mmrestripefs** command may then be invoked to migrate all of the data at once, rather than migrating each file individually. Note that the rebalance (**-b**) option also performs data placement on all files, whereas the placement (**-p**) option rebalances only the files that it moves.

If you do not replicate all of your files, the migrate (**-m**) option is useful to protect against data loss when you have an advance warning that a disk may be about to fail, for example, when the error logs show an excessive number of I/O errors on a disk. Suspending the disk and issuing the **mmrestripefs** command with the **-m** option is the quickest way to migrate only the data that would be lost if the disk failed.

If you do not use replication, the **-m** and **-r** options are equivalent; their behavior differs only on replicated files. After a successful re-replicate (**-r** option) all suspended disks are empty. A migrate operation, using the **-m** option, leaves data on a suspended disk as long as at least one other replica of the data remains on a disk that is not suspended. Restripping a file system includes re-replicating it; the **-b** option performs all the operations of the **-m** and **-r** options.

Consider the necessity of restripping and the current demands on the system. New data which is added to the file system is correctly striped. Restripping a large file system requires extensive data copying and may affect system performance. Plan to perform this task when system demand is low.

If you are sure you want to proceed with the restripe operation:

1. Use the **mmchdisk** command to suspend any disks to which you *do not* want the file system restripped. You may want to exclude disks from file system restripping because they are failing. See “Changing GPFS disk states and parameters” on page 37.
2. Use the **mmfsdisk** command to assure that all disk devices to which you *do* want the file system restripped are in the up/normal state. See “Displaying GPFS disk states” on page 36.

Specify the target file system with the **mmrestripefs** command. For example, to rebalance (**-b** option) file system **fs2** after adding an additional RAID device, enter:

```
mmrestripefs fs2 -b
```

The system displays information similar to:

```
mmrestripefs fs2 -b
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
 48 % complete on Wed Aug 16 16:47:53 2000
 96 % complete on Wed Aug 16 16:47:56 2000
100 % complete on Wed Aug 16 16:47:56 2000
```

```

GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
  98 % complete on Wed Aug 16 16:48:02 2000
 100 % complete on Wed Aug 16 16:48:02 2000
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
GPFS: 6027-552 Scan completed successfully.

```

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

See the “mmrestripefs Command” on page 268 for complete usage information.

Querying file system space

Although you can use the **df** command to summarize the amount of free space on all GPFS disks, the **mmdf** command is useful for determining how well-balanced the file system is across your disks. You can also use the **mmdf** command to diagnose space problems that might result from fragmentation.

Note: The **mmdf** command may require considerable metadata I/O, and should be run when the system load is light.

Specify the file system you want to query with the **mmdf** command. For example, to query available space on all disks in the file system **fs1**, enter:

```
mmdf fs1
```

The system displays information similar to:

disk name	disk size in KB	failure holds group metadata	holds data	free KB in full blocks	free KB in fragments
Disks in storage pool: system					
hd16vsdn10	17793024	-1 yes	yes	17538560 (99%)	1728 (0%)
hd3vsdn01	8880128	2 yes	yes	8658176 (98%)	1600 (0%)
hd4vsdn01	8880128	2 yes	yes	8616448 (97%)	1384 (0%)
hd15vsdn10	17793024	10 yes	yes	17539584 (99%)	1664 (0%)
hd13vsdn02	8880128	4001 yes	yes	8663552 (98%)	1776 (0%)
hd8vsdn01	8880128	4002 yes	yes	8659200 (98%)	1936 (0%)
hd5vsdn01	8880128	4002 yes	yes	8654848 (97%)	1728 (0%)
hd33n09	17796008	4003 yes	yes	17540864 (99%)	2240 (0%)

(pool total)	257800488			252091136 (98%)	46928 (0%)
Disks in storage pool: fs1sp1					
hd30n01	8897968	8 no	yes	8895488 (100%)	424 (0%)
hd31n01	8897968	8 no	yes	8895488 (100%)	424 (0%)

(pool total)	17795936			17790976 (100%)	848 (0%)
=====					
(data)	266716296			261222144 (98%)	44576 (0%)
(metadata)	248920360			243217408 (98%)	46048 (0%)
=====					
(total)	275596424			269882112 (98%)	47776 (0%)

Inode Information

```

Number of used inodes:      9799
Number of free inodes:     4990393
Number of allocated inodes: 5000192
Maximum number of inodes:  5000192

```

See the “mmdf Command” on page 176 for complete usage information.

Querying and reducing file system fragmentation

Disk fragmentation within a file system is an unavoidable condition. When a file is closed after it has been written to, the last logical block of data is reduced to the actual number of subblocks required, thus creating a fragmented block.

In order to **write** to a file system, free full blocks of disk space are required. Due to fragmentation, it is entirely possible to have the situation where the file system is not full, but an insufficient number of free full blocks are available to **write** to the file system. Replication can also cause the copy of the fragment to be distributed among disks in different failure groups. The **mmdefragfs** command can be used to query the current fragmented state of the file system and reduce the fragmentation of the file system.

In order to reduce the fragmentation of a file system, the **mmdefragfs** command migrates fragments to free space in another fragmented disk block of sufficient space, thus creating a free full block. There is no requirement to have a free full block in order to run the **mmdefragfs** command. The execution time of the **mmdefragfs** command depends on the size and allocation pattern of the file system. For a file system with a large number of disks, the **mmdefragfs** command will run through several iterations of its algorithm, each iteration compressing a different set of disks. Execution time is also dependent on how fragmented the file system is. The less fragmented a file system, the shorter time for the **mmdefragfs** command to execute.

The fragmentation of a file system can be reduced on all disks which are not suspended or stopped. If a disk is suspended or stopped, the state of the disk, not the utilization information, will be displayed as output for the **mmdefragfs** command.

The **mmdefragfs** command can be run on both a mounted or an unmounted file system, but achieves best results on an unmounted file system. Running the command on a mounted file system can cause conflicting allocation information and consequent retries to find a new free subblock of the correct size to store the fragment in.

Querying file system fragmentation

To query the current status of the amount of fragmentation for a file system, specify the file system name along with the **-i** option on the **mmdefragfs** command. For example, to display the current fragmentation information for file system **fs0**, enter:

```
mmdefragfs fs0 -i
```

The system displays information similar to:

```
"fs0"    10304 inodes:    457 allocated / 9847 free
```

disk name	disk size in nSubblk	free subblk in full blocks	free subblk in fragments	% free blk	% blk util
gpfs68nsd	4390912	4270112	551	97.249	99.544
gpfs69nsd	4390912	4271360	490	97.277	99.590
(total)	8781824	8541472	1041		99.567

See the “mmdefragfs Command” on page 157 for complete usage information.

Reducing file system fragmentation

You can reduce the amount of fragmentation for a file system by issuing the **mmdefragfs** command, with or without a desired block usage goal. For example, to reduce the amount of fragmentation for file system **fs1** with a goal of 100% utilization, enter:

```
mmdefragfs fs1 -u 100
```

The system displays information similar to:

Defragmenting file system 'fs1'...

Defragmenting until full block utilization is 100.00%, currently 99.96%

```
1 % complete on Wed May 16 15:21:49 2007
6 % complete on Wed May 16 15:21:53 2007
12 % complete on Wed May 16 15:21:56 2007
21 % complete on Wed May 16 15:21:59 2007
32 % complete on Wed May 16 15:22:02 2007
47 % complete on Wed May 16 15:22:05 2007
68 % complete on Wed May 16 15:22:08 2007
91 % complete on Wed May 16 15:22:11 2007
95 % complete on Wed May 16 15:22:30 2007
100 % complete on Wed May 16 15:22:38 2007
```

disk name	free subblk in full			free subblk in		% free blk		% blk util	
	before	after	blk freed	before	after	before	after	before	after
hd16vsdn10	2192320	2192320	0	216	216	98.57	98.57	99.99	99.99
hd3vsdn01	1082272	1082272	0	200	200	97.50	97.50	99.98	99.98
hd4vsdn01	1077056	1077056	0	173	173	97.03	97.03	99.98	99.98
hd20vsdn02	1082496	1082496	0	400	400	97.52	97.52	99.96	99.96
hd6vsdn01	1077120	1077120	0	120	120	97.04	97.04	99.99	99.99
hd7vsdn01	1077344	1077344	0	246	246	97.06	97.06	99.98	99.98
hd2vsdn01	1084032	1084032	0	336	336	97.66	97.66	99.97	99.97
hd9vsdn01	1078272	1078272	0	217	217	97.14	97.14	99.98	99.98
hd10vsdn01	1080000	1080000	0	263	263	97.30	97.30	99.98	99.98
hd11vsdn02	1109216	1109216	0	110	110	99.93	99.93	99.99	99.99
hd18vsdn10	2196544	2196544	0	306	306	98.76	98.76	99.99	99.99
hd18vsdn02	1083168	1083168	0	246	246	97.58	97.58	99.98	99.98
hd24n01	1079616	1079616	0	200	200	97.07	97.07	99.98	99.98
hd25n01	1079808	1079808	0	229	229	97.08	97.08	99.98	99.98
hd26n01	1085056	1085056	0	205	205	97.56	97.56	99.98	99.98
hd27n01	1085408	1085408	0	237	237	97.59	97.59	99.98	99.98
hd28n01	1085312	1085312	0	193	193	97.58	97.58	99.98	99.98
hd29n01	1085792	1085792	0	260	260	97.62	97.62	99.98	99.98
hd30n01	1111936	1111936	0	53	53	99.97	99.97	100.00	100.00
hd31n01	1111936	1111936	0	53	53	99.97	99.97	100.00	100.00
hd21n01	1078912	1078912	0	292	292	97.00	97.00	99.97	99.97
hd23n01	1079392	1079392	0	249	249	97.05	97.05	99.98	99.98
hd15vsdn10	2192448	2192448	0	208	208	98.58	98.58	99.99	99.99
hd13vsdn02	1082944	1082944	0	222	222	97.56	97.56	99.98	99.98
hd8vsdn01	1082400	1082400	0	242	242	97.51	97.51	99.98	99.98
hd5vsdn01	1081856	1081856	0	216	216	97.46	97.46	99.98	99.98
hd33n09	2192608	2192608	0	280	280	98.57	98.57	99.99	99.99
(total)	33735264	33735264	0	5972	5972			99.98	99.98

Defragmentation complete, full block utilization is 99.96%.
Re-issue command to try to reach target utilization of 100.00%.

See the “mmdefragfs Command” on page 157 for complete usage information.

Backing up a file system

The **mmbackup** command enables root users to backup the files of a GPFS file system to storage using a Tivoli® Storage Manager (TSM) server. Once a file system has been backed up, the user can restore files using the interfaces provided by TSM.

The **mmbackup** command provides either:

- A full backup of all files in the file system.
- An incremental backup of only those files which have changed or been deleted since the last backup. Files that have changed since the last backup are updated in the backup, and files that have been deleted since the last backup are expired from the backup.

Required level of Tivoli Storage Manager

When using the GPFS **mmbackup** command, Tivoli Storage Manager must be at Version 4 Release 2 (4.2) or later.

Setting up TSM for use by GPFS

The **mmbackup** command requires a working TSM environment:

1. The TSM server code must be installed and the TSM server must be configured for backup on the TSM server node specified in the file on the **-n ControlFile** option.
2. The TSM client code must be installed on each client node specified in the file on the **-n ControlFile** option. The lines changed in the **dsm.opt** and **dsm.sys** files to enable **mmbackup** to work with TSM must be changed in these files on all of the TSM client nodes used by **mmbackup**. For example, the **dsm.opt** file may be:

```
SErvername k164n06.kgn.ibm.com
```

The **dsm.sys** file may be:

```
SErvername k164n06.kgn.ibm.com  
COMMethod TCPip  
TCPPort 1500  
TCPServeraddress 9.118.95.25  
PASSWORDAccess generate  
NODEname GPFSnodes
```

3. The TSM clients must be configured to communicate with the TSM server.
4. TSM must be made aware that the various TSM clients are all working on the same file system, not different file systems having the same name on different client machines. This is accomplished by coding the same value for the **nodename** keyword in the **dsm.sys** file located in the Tivoli client directory (**/usr/tivoli/tsm/client/ba/bin** for AIX, **/opt/tivoli/tsm/client/ba/bin** for Linux) on each client.
5. Restoration of backed up data must be done using TSM interfaces. This can be done with the client command line interface or the TSM web client. The TSM web client interface must be made operational if you desire to use this interface for restoring data to the file system from the TSM server.

Attention: If you are using the TSM Backup Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

You can view or download the TSM documentation at the IBM Tivoli Storage Manager Info Center at <http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp?toc=/com.ibm.itstorage.doc/toc.xm>

Specifying TSM parameters with the **mmbackup** command

Control of the backup is accomplished by means of parameters in the control file which is specified using the **-n** option on the **mmbackup** command. The parameters that may be specified in this control file are the TSM backup server node, the TSM backup client nodes, and the number of backup processes per client. The backup server node must be specified, as well as at least one backup client node. Specifying multiple backup client nodes and a larger number of backup processes per client allows for greater parallelism and performance.

- | For a discussion of the files created and maintained by the **mmbackup** command, see the topic *GPFS backup data* in the *General Parallel File System: Concepts, Planning, and Installation Guide*.

Using APIs to develop backup applications

IBM has supplied a set of subroutines that are useful to create backups or collect information about all files in a file system. Each subroutine is described in Chapter 9, “GPFS programming interfaces,” on page 291. These subroutines are more efficient for traversing a file system, and provide more features than the standard POSIX interfaces. These subroutines operate on a snapshot or on the active file system. They have the ability to return all files, or only files that have changed since some earlier snapshot, which is useful for incremental backup.

A typical use of these subroutines is the following scenario:

1. Create a snapshot using the **mmcrsnapshot** command. For more information on snapshots, see *General Parallel File System: Advanced Administration Guide*.
2. Open an inode scan on the snapshot using the **gpfs_open_inodescan()** subroutine.
3. Retrieve inodes using the **gpfs_next_inode()** subroutine.
4. Read the file data:
 - a. Open the file using the **gpfs_iopen()** subroutine.
 - b. Read the file using the **gpfs_iread()**, **gpfs_ireadx()**, or **gpfs_ireaddir()** subroutines.
 - c. Close the file using the **gpfs_iclose()** subroutine.

The **gpfs_ireadx()** subroutine is more efficient than **read()** or **gpfs_iread()** for sparse files and for incremental backups. The **gpfs_ireaddir()** subroutine is more efficient than **readdir()**, because it returns file type information. There are also subroutines for reading symbolic links, **gpfs_ireadlink()** and for accessing file attributes, **gpfs_igetattr()**.

IBM has supplied a backup application program, **tsbackup.C**, to serve as an example. This example is located in **/usr/lpp/mmfs/samples/util** and consists of these files:

- **tsbackup.C** - A utility for backing up a GPFS file system to a TSM server using TSM clients.
- **tsbackup.h** - A header file containing necessary declarations.

The **tsbackup** sample application is invoked by the **mmbackup** script in **/usr/lpp/mmfs/bin**, and it invokes the **mmexecsmcmd** script in that same directory to drive TSM clients. These scripts may also be examined as examples.

Chapter 4. Managing disks

With file systems created with GPFS 2.3 or later, the theoretical limit on the maximum number of disks in a file system has been increased from 4096 to approximately 268 million. However, the actual limit enforced by the current version of GPFS is 2048. It can be increased if necessary (contact IBM to discuss increasing the limit).

Note: A LUN provided by a storage subsystem is a disk for the purposes of this documentation, even if the LUN is made up of multiple physical disks.

The disks may have connectivity to each node in the cluster, be managed by network shared disk servers, or a combination of the two. See the **mmcrnsd** command and the *General Parallel File System: Concepts, Planning, and Installation Guide*, and search on *network shared disk creation considerations*.

The disk related tasks performed on a GPFS file system include:

1. "Displaying disks in a GPFS cluster"
2. "Adding disks to a file system" on page 32
3. "Deleting disks from a file system" on page 33
4. "Replacing disks in a GPFS file system" on page 34
5. "Additional considerations for managing disks" on page 35
6. "Displaying GPFS disk states" on page 36
7. "Changing GPFS disk states and parameters" on page 37
8. "Changing your NSD configuration" on page 38
9. "Changing NSD server usage and failback" on page 39
10. "Enabling and disabling Persistent Reserve" on page 40

Displaying disks in a GPFS cluster

You can display the disks that belong to your GPFS cluster by issuing the **mmllnsd** command.

The default is to display information for all disks defined to the cluster (**-a**). Otherwise, you may choose to display the information for a particular file system (**-f**) or for all disks which do not belong to any file system (**-F**).

To display the default information for all of the NSDs belonging to the cluster, enter:

```
mmllnsd
```

The system displays information similar to:

File system	Disk name	NSD servers
fs2	hd3n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd4n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd5n98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd6n98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd7vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd8vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd9vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd10vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd11vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	hd12vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	sdbnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdcnsc	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sddnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com

fs2	sdensd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdgnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdfnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdhnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
(free disk)	hd2n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com

To find out the local device names for the disks, use the **mmlsnsd** command with the **-m** option. For example, issuing **mmlsnsd -m** produces output similar to this:

Disk name	NSD volume ID	Device	Node name	Remarks
hd10vsdn98	0972846245C8E93C	/dev/hd10vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd10vsdn98	0972846245C8E93C	/dev/hd10vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd11vsdn98	0972846245C8E93F	/dev/hd11vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd11vsdn98	0972846245C8E93F	/dev/hd11vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd12vsdn98	0972846245C8E941	/dev/hd12vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd12vsdn98	0972846245C8E941	/dev/hd12vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd2n97	0972846145C8E924	/dev/hdisk2	c5n97g.ppd.pok.ibm.com	server node
hd2n97	0972846145C8E924	/dev/hdisk2	c5n98g.ppd.pok.ibm.com	server node
hd3n97	0972846145C8E927	/dev/hdisk3	c5n97g.ppd.pok.ibm.com	server node
hd3n97	0972846145C8E927	/dev/hdisk3	c5n98g.ppd.pok.ibm.com	server node
hd4n97	0972846145C8E92A	/dev/hdisk4	c5n97g.ppd.pok.ibm.com	server node
hd4n97	0972846145C8E92A	/dev/hdisk4	c5n98g.ppd.pok.ibm.com	server node
hd5n98	0972846245EB501C	/dev/hdisk5	c5n97g.ppd.pok.ibm.com	server node
hd5n98	0972846245EB501C	/dev/hdisk5	c5n98g.ppd.pok.ibm.com	server node
hd6n98	0972846245DB3AD8	/dev/hdisk6	c5n97g.ppd.pok.ibm.com	server node
hd6n98	0972846245DB3AD8	/dev/hdisk6	c5n98g.ppd.pok.ibm.com	server node
hd7vsdn97	0972846145C8E934	/dev/hd7vsdn97	c5n97g.ppd.pok.ibm.com	server node

Adding disks to a file system

Many file systems grow rapidly so you might decide that more disk space is required for your file system soon after creation. Before adding disks, see *General Parallel File System: Concepts, Planning, and Installation Guide* for details on creating disk descriptors.

- | Storage in a file system is divided in storage pools. The maximum size of any one disk that can be added to an existing storage pool is set approximately to the sum of the disk sizes when the storage pool is created. The actual value is shown in the **mmdf** command output.
- | Once a storage pool is created, the maximum size *cannot* be altered. However, you can create a new pool with larger disks, and then move data from the old pool to the new one.
- | When establishing a storage pool and when adding disks later to an existing storage pool, you should try to keep the sizes of the disks fairly uniform. GPFS allocates blocks round robin, and as the utilization level rises on all disks, the small ones will fill up first and all files created after that will be spread across fewer disks, which reduces the amount of prefetch that can be done for those files.

To add disks to a GPFS file system, you first must decide if you will:

1. Create new disks using the **mmcrnsd** command.
You should also decide whether to use the rewritten disk descriptor file produced by the **mmcrnsd** command, or create a new list of disk descriptors. When using the rewritten file, the *Disk Usage* and *Failure Group* specifications remain the same as specified on the **mmcrnsd** command.
2. Select disks no longer in use in any file system. Issue the **mmlsnsd -F** command to display the available disks.

The disk may then be added to the file system using the **mmadddisk** command.

For example, to add the disk **gpfs2nsd** to the file system **fs2**, have it join failure group **1** in the storage pool **system**, and rebalance the existing files to take advantage of the added space, enter:

```
mmadddisk fs2 gpfs2nsd:::1 -r
```

The system displays information similar to:

```
GPFS: 6027-531 The following disks of fs2 will be formatted on node k145n01:
gpfs2nsd: size 4390912 KB
Extending Allocation Map
GPFS: 6027-1503 Completed adding disks to file system fs2.
mmadddisk: 6027-1371 Propagating the changes to all affected nodes.
This is an asynchronous process.
Restriping fs2 ...
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
  70 % complete on Wed Aug 16 15:14:28 2000
 100 % complete on Wed Aug 16 15:14:29 2000
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
  6 % complete on Wed Aug 16 15:14:45 2000
 100 % complete on Wed Aug 16 15:14:46 2000
GPFS: 6027-552 Scan completed successfully.
Done
```

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

When using an IBM eServer™ High Performance Switch (HPS) in your configuration, it is suggested you process your disks in two steps:

1. Create virtual shared disks on each physical disk with the **mmcrvsd** command.
2. Using the rewritten disk descriptors from the **mmcrvsd** command, create NSDs with the **mmcrnsd** command.

Deleting disks from a file system

Before deleting a disk use the **mmdf** command to determine whether there is enough free space on the remaining disks to store the file system.

Note: See “Querying file system space” on page 25 for more information about diagnosing space problems.

Consider how fragmentation may increase your storage requirements, especially when the file system contains a large number of small files. A margin of 150 percent of the size of the disks being deleted should be sufficient to allow for fragmentation when small files predominate. For example, in order to delete a 4 GB disk from your file system, which contains user home directories with small files, you should first determine that the other disks in the file system contain a total of 6 GB of free space.

If you do not replicate your file system data, you should rebalance the file system using the **mmrestripefs -b** command. If you replicate your file system data, run the **mmrestripefs -r** command after the disk has been deleted. This ensures that all data will still exist with correct replication after the disk is deleted. The **mmdeldisk** command only migrates data that would otherwise be lost, not data that will be left in a single copy.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

Do not delete stopped disks, if at all possible. Start any stopped disk before attempting to delete it from the file system. If the disk cannot be started you will have to consider it permanently damaged. You will need to delete the disk using the appropriate options. If metadata was stored on the disk, you will need to execute the offline version of the **mmfsck** command. See the *General Parallel File System: Problem Determination Guide* and search for *NSD failures* for further information on handling this.

When deleting disks from a file system, the disks may or may not be available. If the disks being deleted are still available, GPFS moves all of the data from those disks to the disks remaining in the file system. However, if the disks being deleted are damaged, either partially or permanently, it is not possible to move all of the data and you will receive I/O errors during the deletion process. For instructions on how to handle damaged disks, see the *General Parallel File System: Problem Determination Guide* and search for *Disk media failure*.

Specify the file system and the names of one or more disks to delete with the **mmdeldisk** command. For example, to delete the disk **hd2n97** from the file system **fs1** enter:

```
mmdeldisk fs2 hd2n97
```

The system displays information similar to:

```
Deleting disks ...
Scanning 'system' storage pool
Scanning file system metadata, phase 1 ...
  11 % complete on Fri Feb  3 16:01:36 2006
  31 % complete on Fri Feb  3 16:01:42 2006
  86 % complete on Fri Feb  3 16:01:58 2006
 100 % complete on Fri Feb  3 16:02:01 2006
Scan completed successfully.
Scanning file system metadata, phase 2 ...
  40 % complete on Fri Feb  3 16:02:04 2006
 100 % complete on Fri Feb  3 16:02:08 2006
Scan completed successfully.
Scanning file system metadata, phase 3 ...
   9 % complete on Fri Feb  3 16:02:11 2006
  42 % complete on Fri Feb  3 16:02:18 2006
 100 % complete on Fri Feb  3 16:02:22 2006
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
   1 % complete on Fri Feb  3 16:02:32 2006
  12 % complete on Fri Feb  3 16:02:39 2006
  23 % complete on Fri Feb  3 16:02:47 2006
  51 % complete on Fri Feb  3 16:02:55 2006
  82 % complete on Fri Feb  3 16:03:01 2006
 100 % complete on Fri Feb  3 16:03:14 2006
Scan completed successfully.
tsdeldisk64 completed.
mmdeldisk: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Refer to “mmdeldisk Command” on page 162 for syntax and usage information.

Replacing disks in a GPFS file system

Replacing an existing disk in a GPFS file system with a new one is the same as performing a delete disk operation followed by an add disk. However, this operation eliminates the need to restripe the file system following the separate delete disk and add disk operations as data is automatically moved to the new disk.

Under no circumstances should you replace a stopped disk. You need to start a stopped disk before replacing it. See the *General Parallel File System: Problem Determination Guide* and search for *disk media failure* for further information on handling this.

When replacing disks in a GPFS file system, you must first decide if you will:

1. Create new disks using the **mmcrnsd** command.

You should also decide whether to use the rewritten disk descriptor file produced by the **mmcrnsd** command, or create a new list of disk descriptors. When using the rewritten file, the *Disk Usage* and *Failure Group* specifications will remain the same as specified on the **mmcrnsd** command.

2. Select NSDs no longer in use by another GPFS file system. Issue the **mmisnsd -F** command to display the available disks.

To replace a disk in the file system, use the **mmrpldisk** command. For example, to replace the NSD **hd3n97** in file system **fs2** with the existing NSD **hd2n97**, which is no longer in use by another file system, enter:

```
mmrpldisk fs2 hd3n97 hd2n97
```

The system displays information similar to:

Replacing hd3n97 ...

The following disks of fs2 will be formatted on node k145n03:

hd2n97: size 142028570 KB

Extending Allocation Map

0 % complete on Fri Feb 3 16:12:49 2006

1 % complete on Fri Feb 3 16:12:54 2006

40 % complete on Fri Feb 3 16:14:59 2006

91 % complete on Fri Feb 3 16:17:10 2006

100 % complete on Fri Feb 3 16:17:30 2006

Completed adding disks to file system fs2.

Scanning 'system' storage pool

Scanning file system metadata, phase 1 ...

2 % complete on Fri Feb 3 16:17:44 2006

10 % complete on Fri Feb 3 16:17:47 2006

62 % complete on Fri Feb 3 16:18:06 2006

78 % complete on Fri Feb 3 16:18:13 2006

100 % complete on Fri Feb 3 16:18:19 2006

Scan completed successfully.

Scanning file system metadata, phase 2 ...

67 % complete on Fri Feb 3 16:18:25 2006

100 % complete on Fri Feb 3 16:18:26 2006

Scan completed successfully.

Scanning file system metadata, phase 3 ...

22 % complete on Fri Feb 3 16:18:29 2006

40 % complete on Fri Feb 3 16:18:36 2006

74 % complete on Fri Feb 3 16:18:49 2006

100 % complete on Fri Feb 3 16:18:56 2006

Scan completed successfully.

Scanning file system metadata, phase 4 ...

Scan completed successfully.

Scanning user file metadata ...

4 % complete on Fri Feb 3 16:19:00 2006

26 % complete on Fri Feb 3 16:19:07 2006

100 % complete on Fri Feb 3 16:19:31 2006

Scan completed successfully.

Done

mmrpldisk: Propagating the cluster configuration data to all affected nodes. This is an asynchronous process.

Additional considerations for managing disks

If you delete, replace, or suspend a disk with strict replication enforced, you may receive an ENOSPC error when you create or append data to an existing file.

If you need to delete, replace, or suspend a disk and you need to write new data while the disk is offline, you can disable strict replication before you perform the disk action. However, data written while replication

is disabled will not be properly replicated. Therefore, after you perform the disk action, you must re-enable strict replication and run the **mmrestripe -r** command. To determine if a file system has strict replication enforced, issue the **mmlsfs -K** command.

Displaying GPFS disk states

You can display the current state of one or more disks in your file system by issuing the **mmlsdisk** command.

The information includes parameters that were specified on the **mmcrfs** command, and the current availability and status of the disks. For example, to display the current status of the disk **hd8vsdn100** in the file system **fs1**, enter:

```
mmlsdisk fs1 -d hd8vsdn100
```

Status is displayed in a format similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
hd8vsdn100	nsd	512	1	no	yes	ready	up	sp1

Refer to “mmlsdisk Command” on page 213 for syntax and usage information.

Disk availability

A disk’s availability determines whether GPFS is able to read and write to the disk. There are four possible values for availability:

up The disk is available to GPFS for normal **read** and **write** operations.

down No **read** and **write** operations can be performed on the disk.

recovering

An intermediate state for disks coming up, during which GPFS verifies and corrects data. **read** operations can be performed while a disk is in this state but **write** operations cannot.

unrecovered

Not all disks were successfully brought up.

Disk availability is automatically changed from **up** to **down** when GPFS detects repeated I/O errors. You can also change the availability of a disk by issuing the **mmchdisk** command.

Disk status

Disk status controls data placement and migration. Status changes as a result of a pending delete operation, or when the **mmchdisk** command is issued to allow file rebalancing or re-replicating prior to disk replacement or deletion.

Disk status has five possible values, but three are transitional:

ready Normal status.

suspended

Indicates that data is to be migrated off this disk.

being emptied

Transitional status in effect while a disk deletion is pending.

replacing

Transitional status in effect for old disk while replacement is pending.

replacement

Transitional status in effect for new disk while replacement is pending.

GPFS allocates space only on disks with a status of **ready** or **replacement**.

GPFS migrates data off of disks with a status of **being emptied**, **replacing**, or **suspended**, onto disks with a status of **ready** or **replacement**. During disk deletion or replacement, data is automatically migrated as part of the operation. The **mmrestripefs** command must be issued to initiate data migration from a suspended disk.

See “Deleting disks from a file system” on page 33, “Replacing disks in a GPFS file system” on page 34, and “Restripping a GPFS file system” on page 24.

Changing GPFS disk states and parameters

You might find it necessary to change a disk’s state if there is some indication of disk failure or if you need to restripe the file system.

Refer to “Displaying GPFS disk states” on page 36 for a detailed description of disk states. You can change both the availability and status of a disk using the **mmchdisk** command:

- Change disk availability using the **mmchdisk** command and the **stop** and **start** options
- Change disk status using the **mmchdisk** command and the **suspend** and **resume** options.

Issue the **mmchdisk** command with one of the following four options to change disk state:

suspend

Instructs GPFS to stop allocating space on the specified disk. Place a disk in this state prior to disk deletion or replacement. This is a user-initiated state that GPFS will never use without an explicit command to change disk state.

Note: A disk remains suspended until it is explicitly resumed. Restarting GPFS or rebooting nodes does not restore normal access to a suspended disk.

resume

Informs GPFS that a disk previously suspended is now available for allocating new space. Resume a disk only when you’ve suspended it and decided not to delete or replace it. If the disk is currently in a stopped state, it remains stopped until you specify the **start** option. Otherwise, normal **read** and **write** access to the disk resumes.

stop Instructs GPFS to stop any attempts to access the specified disk. Use this option to inform GPFS that a disk has failed or is currently inaccessible because of maintenance. A disk’s availability remains **down** until it is explicitly started with the **start** option.

start Informs GPFS that a disk previously stopped is now accessible. GPFS does this by first changing the disk availability from **down** to **recovering**. The file system metadata is then scanned and any missing updates (replicated data that was changed while the disk was down) are repaired. If this operation is successful, the availability is then changed to **up**.

If the metadata scan fails, availability is set to **unrecovered**. This could occur if other disks remain in **recovering** or an I/O error has occurred. Repair all disks and paths to disks. See **mmfsck**. The metadata scan can then be re-initiated at a later time by issuing the **mmchdisk start** command again.

If more than one disk in the file system is down, they should all be started at the same time by using the **-a** option. If you start them separately and metadata is stored on any disk that remains down, the **mmchdisk start** command fails.

For example, to suspend the **hd8vsdn100** disk in the file system **fs1**, enter:

```
mmchdisk fs1 suspend -d hd8vsdn100
```

To confirm the change, enter:

```
mm1sdisk fs1 -d hd8vsdn100
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
hd8vsdn100	nsd	512	7	yes	yes	suspended	up	system

You can also use the **mmchdisk** command with the **change** option to change the *Disk Usage* and *Failure Group* parameters for one or more disks in a GPFS file system. This can be useful in situations where, for example, a file system that contains only RAID disks is being upgraded to add conventional disks that are better suited to storing metadata. After adding the disks using the **mmadddisk** command, the metadata currently stored on the RAID disks would have to be moved to the new disks to achieve the desired performance improvement. To accomplish this, first the **mmchdisk change** command would be issued to change the *Disk Usage* parameter for the RAID disks to **dataOnly**. Then the **mmrestripefs** command would be used to restripe the metadata off the RAID device and onto the conventional disks.

For example, to specify that metadata should no longer be stored on disk **hd8vsdn100**, enter:

```
mmchdisk fs1 change -d "hd8vsdn100:::dataOnly"
```

To confirm the change, enter:

```
mm1sdisk fs1 -d hd8vsdn100
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
hd8vsdn100	nsd	512	1	no	yes	ready	up	sp1

See the “mmchdisk Command” on page 100 and the “mm1sdisk Command” on page 213 for complete usage information.

Changing your NSD configuration

Once your NSDs have been created (see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the **mmcrnsd** command), you may change the configuration attributes as your system requirements change.

By issuing the **mmchnsd** command you can:

- Specify up to eight servers for an NSD that does not have one.
- Change the NSD server nodes specified in the server list.
- Delete the server list. The disk must now be SAN-attached to all nodes in the cluster on which the file system will be mounted.

You must follow these rules when changing NSDs:

- You must identify the disks by the NSD names that were given to them by the **mmcrnsd** command.
- You must explicitly specify values for all NSD servers in the list, even if you are only changing one of the values.
- The file system that contains the NSD being changed must be unmounted prior to issuing the **mmchnsd** command.
- The NSD must be properly connected to the new nodes prior to issuing the **mmchnsd** command.
- This command cannot be used to change the *DiskUsage* or *FailureGroup* for an NSD. You must issue the **mmchdisk** command to change these.
- You cannot change the name of the NSD.

For example, to assign node **k145n07** as an NSD server for disk **gpfs47nsd**:

1. Make sure that **k145n07** is not already assigned to the server list by issuing the **mmlsnsd** command.

```
mmlsnsd -d "gpfs47nsd"
```

The system displays information similar to:

File system	Disk name	NSD server nodes
fs1	gpfs47nsd	k145n09

2. Unmount the file system on all nodes and ensure that the disk is connected to the new node (**k145n07**).

3. Issue the **mmchnsd** command:

```
mmchnsd "gpfs47nsd:k145n09,k145n07"
```

4. Verify the changes by issuing the **mmlsnsd** command:

```
mmlsnsd -d gpfs47nsd
```

The system displays information similar to:

File system	Disk name	NSD servers
fs2	gpfs47nsd	k145n09.ppd.pok.ibm.com,k145n07.ppd.pok.ibm.com

Changing NSD server usage and failback

GPFS determines if a node has physical or virtual connectivity to an underlying NSD disk through a sequence of commands invoked from the GPFS daemon. This determination is called *disk discovery* and occurs at both initial GPFS startup as well as whenever a file system is mounted.

The default order of access used in disk discovery:

1. Local **/dev** block device interfaces for virtual shared disk, SAN, SCSI or IDE disks
2. NSD servers

The **useNSDserver** file system mount option can be used to set the order of access used in disk discovery, and limit or eliminate switching from local access to NSD server access, or the other way around. This option is specified using the **-o** flag of the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands, and has one of these values:

always

Always access the disk using the NSD server.

asfound

Access the disk as found (the first time the disk was accessed). No change of disk access from local to NSD server, or the other way around, is performed by GPFS.

asneeded

Access the disk any way possible. This is the default.

never Always use local disk access.

For example, to always use the NSD server when mounting file system **fs1**, issue this command:

```
mmmount fs1 -o useNSDserver=always
```

To change the disk discovery of a file system that is already mounted: cleanly unmount it, wait for the unmount to complete, and then mount the file system using the desired **-o useNSDserver** option.

Enabling and disabling Persistent Reserve

GPFS can use Persistent Reserve (PR) functionality to improve failover times (with some restrictions).

The following restrictions apply to the use of PR:

- | • PR is only supported on AIX nodes. This means that:
 - | – If the disks have defined NSD servers, then all NSD server nodes must be running AIX.
 - | – If the disks are SAN-attached to all nodes, then all nodes in the cluster must be running AIX.
- | • PR is only supported on NSDs built directly on hdisks.
- | • The disk subsystems must support PR
- | • GPFS supports a mix of PR disks and other disks. However, you will only realize improved failover times if **all** disks in the cluster support PR.
- | • GPFS only supports PR in the local cluster. Remote mounts must access the disks through an NSD server.
- | • When you enable or disable PR, you must stop GPFS on all nodes.

To enable (or disable) Persistent Reserve, issue the command:

```
mmchconfig usePersistentReserve={yes|no}
```

For fast recovery times with Persistent Reserve, you should also set the *failureDetectionTime* configuration parameter. For fast recovery, a recommended value would be 10. You can set this by issuing the command:

```
mmchconfig failureDetectionTime=10
```

To determine if the disks on the servers and the disks of a specific node have PR enabled, issue the following command from the node:

```
mmfnsd -X
```

The system responds with something similar to:

Disk name	NSD volume ID	Device	Devtype	Node name	Remarks
gpfs10nsd	09725E5E43035A99	/dev/hdisk6	hdisk	k155n14.kgn.ibm.com	server node,pr=yes
gpfs10nsd	09725E5E43035A99	/dev/hdisk8	hdisk	k155n16.kgn.ibm.com	server node,pr=yes
gpfs10nsd	09725E5E43035A99	/dev/hdisk6	hdisk	k155n17.kgn.ibm.com	directly attached pr=yes

If the GPFS daemon has been started on all the nodes in the cluster and the file system has been mounted on all nodes that have direct access to the disks, then pr=yes should be on all hdisks. If you do not see this, there is a problem. Refer to the *General Parallel File System: Problem Determination Guide* for additional information on Persistent Reserve errors.

Chapter 5. Managing GPFS quotas

The GPFS quota system helps you to control the allocation of files and data blocks in a file system.

GPFS quotas can be defined for:

- Individual users
- Groups of users
- Individual filesets

Quotas are enabled by the system administrator when control over the amount of space used by the individual users, groups of users, or individual filesets is required.

Quota related tasks include:

1. “Enabling and disabling GPFS quota management”
2. “Explicitly establishing and changing quotas” on page 42
3. “Default quotas” on page 42
4. “Checking quotas” on page 43
5. “Listing quotas” on page 44
6. “Activating quota limit checking” on page 45
7. “Deactivating quota limit checking” on page 45
8. “Creating file system quota reports” on page 46

For GPFS fileset quotas, see the section *Filesets* in *General Parallel File System: Advanced Administration Guide*.

Enabling and disabling GPFS quota management

You can enable GPFS quota management on new or existing GPFS file systems, establish quota values, and disable quota management by following the steps in this topic.

To enable GPFS quota management on a new GPFS file system:

1. Specify the **-Q yes** option on the **mmcrfs** command. This option automatically activates quota enforcement whenever the file system is mounted.
2. Mount the file system.
3. Issue the **mmedquota** command to explicitly set quota values for users, groups, or filesets. See “Explicitly establishing and changing quotas” on page 42.

To enable GPFS quota management on an existing GPFS file system:

1. Unmount the file system everywhere.
2. Run the **mmchfs -Q yes** command. This command automatically activates quota enforcement whenever the file system is mounted.
3. Remount the file system, activating the new quota files. All subsequent mounts follow the new quota setting.
4. Compile inode and disk block statistics using the **mmcheckquota** command. See “Checking quotas” on page 43. The values obtained can be used to establish realistic quota values when issuing the **mmedquota** command.
5. Issue the **mmedquota** command to explicitly set quota values for users, groups, or filesets. See “Explicitly establishing and changing quotas” on page 42.

Once GPFS quota management has been enabled, you may establish quota values by:

- Setting default quotas for all new users, groups of users, or filesets.
- Explicitly establishing or changing quotas for users, groups of users, or filesets.
- Using the **gpfs_quotactl()** subroutine.

To disable quota management:

1. Unmount the file system everywhere.
2. Run the **mmchfs -Q no** command.
3. Remount the file system, deactivating the quota files. All subsequent mounts obey the new quota setting.

See the “mmcheckquota Command” on page 104, the “mmchfs Command” on page 109, the “mmcrfs Command” on page 130, and the “mmedquota Command” on page 182 for complete usage information. For additional information on quotas, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.

Default quotas

Applying default quotas provides for minimum quota limits for all new users, groups of users, or filesets for a file system. If default quota values for a file system are not enabled, a new user, group of users or fileset for that file system has a quota limit of zero, which establishes no limit to the amount of space that they can use.

To enable default quota values for a file system:

1. The file system must have been created or changed with the **-Q yes** option. See the **mmcrfs** and **mmchfs** commands.
2. Default quotas are enabled with the **mmdefquotaon** command.
3. Default quota values are established for new users, groups, and filesets for a file system, by issuing the **mmdefquota** command.

To apply different quota values for a particular user, group, or fileset, the system administrator must explicitly configure those values using the **mmedquota** command. If after explicit quotas for a user, group, or filesets have been established, it is necessary to reapply the default limits for that user, group, or fileset, you must issue the **mmedquota -d** command.

The default quotas can be deactivated by issuing the **mmdefquotaoff** command.

For example, to create default quotas for users of the file system **fs0**, enter:

```
mmdefquota -u fs0
```

A prompt in your default editor appears:

```
*** Edit quota limits for USR DEFAULT entry
NOTE: block limits will be rounded up to the next multiple of the block size.
block units may be: K, M, or G.
fs1: blocks in use: 0K, limits (soft = 0K, hard = 0K)
inodes in use: 0, limits (soft = 0, hard = 0)
```

See the “mmdefquota Command” on page 150, “mmdefquotaoff Command” on page 152, “mmdefquotaon Command” on page 154, and “mmedquota Command” on page 182 commands for complete usage information.

Explicitly establishing and changing quotas

Use the **mmedquota** command to explicitly establish or change file system quota limits for users, groups of users, or filesets.

When setting quota limits for a file system, replication within the file system should be considered. GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmfsquota** command and the **mmrepquota** command are double the value reported by the **ls** command.

The **mmedquota** command opens a session using your default editor, and prompts you for soft and hard limits for inodes and blocks. For example, to set user quotas for user **jesmith**, enter:

```
mmedquota -u jesmith
```

A prompt in your default editor appears:

```
*** Edit quota limits for USR jesmith:
NOTE: block limits will be rounded up to the next multiple block size.
block units may be: K, M, or G.
gpfs0: blocks in use (in KB): 24576, limits (soft = 0, hard = 0)
inodes in use: 3, limits (soft = 0, hard = 0)
```

Note: A zero quota limit indicates **no** quota limits have been established.

The current (in use) inode and block usage is for display only; it cannot be changed. When establishing a new quota, zeros appear as limits. Replace the zeros, or old values if you are changing existing limits with values based on the user's needs and the resources available. When you close the editor, GPFS checks the values and applies them. If a value which is not valid is specified, GPFS generates an error message. If this occurs, reenter the **mmedquota** command.

You may find it helpful to maintain a *quota prototype*, a set of limits that you can apply by name to any user, group, or fileset without entering the individual values manually. This makes it easy to set the same limits for all. The **mmedquota** command includes the **-p** option for naming a prototypical user, group, or fileset on which limits are to be based.

For example, to set group quotas for all users in a group named **blueteam** to the prototypical values established for **prototeam**, issue:

```
mmedquota -g -p prototeam blueteam
```

You may also reestablish default quotas for a specified user, group of users, or fileset when using the **-d** option on the **mmedquota** command.

See the “mmedquota Command” on page 182 for complete usage information.

Checking quotas

The **mmcheckquota** command counts inode and space usage for a file system and writes the collected data into quota files.

You must use the **mmcheckquota** command if:

1. Quota information is lost due to node failure.
Node failure could leave users unable to open files or deny them disk space that their quotas should allow.
2. The *in doubt* value approaches the quota limit. To see the *in doubt* value, use the **mmfsquota** or **mmrepquota** commands.

As the sum of the *in doubt* value and the current usage may not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset may be constrained by the *in doubt* value. Should the *in doubt* value approach a significant percentage of the quota, use the **mmcheckquota** command to account for the lost space and files.

When issuing the **mmcheckquota** command on a mounted file system, negative *in doubt* values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

During the normal operation of file systems with quotas enabled (not running **mmcheckquota** online), the usage data reflects the actual usage of the blocks and inodes in the sense that if you delete files you should see the usage amount decrease. The *in doubt* value does not reflect how much the user has used already, it is just the amount of quotas that the quota server has assigned to its clients. The quota server does not know whether the assigned amount has been used or not. The only situation where the *in doubt* value is important to the user is when the sum of the usage and the *in doubt* value is greater than the user's quota hard limit. In this case, the user is not allowed to allocate more blocks or inodes unless he brings the usage down.

For example, to check quotas for the file system **fs1** and report differences between calculated and recorded disk quotas, enter:

```
mmcheckquota -v fs1
```

The information displayed shows that the quota information for **USR7** was corrected. Due to a system failure, this information was lost at the server, which recorded 0 subblocks and 0 files. The current usage data counted is 96 subblocks and 3 files. This is used to update the quota:

```
/dev/fs1: quota-check found following differences:  
USR7: 96 subblocks counted (was 0); 3 inodes counted (was 0)
```

See the “mmcheckquota Command” on page 104 for complete usage information.

Listing quotas

The **mmlsquota** command displays the file system quota limits, default quota limits, and current usage information.

GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmlsquota** command and the **mmrepquota** command and the values used to determine if quota limits have been exceeded will be double the value reported by the **ls** command. When issuing the **mmlsquota** command on a mounted file system, negative *in doubt* values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

Specify the quota information for one user, group of users, or fileset with the **mmlsquota** command. If neither **-g**, nor **-u**, nor **-j** is specified, quota data is displayed for the user who entered the command.

For example, to display default quota information for users of all the file systems in the cluster, enter:

```
mmlsquota -d -u
```

Information similar to this is returned:

Filesystem	type	Default Block Limits		Default File Limits		
		quota	limit	quota	limit	Remarks
fs1	USR	no default limits				

This output shows for file system **fs1** a default quota limit of 10240K for users has been established. For file systems **fs2** and **fs3** no default quotas for users has been established.

If you issue the **mmlsquota** command with the **-e** option, the quota system collects updated information from all nodes before returning output. If the node to which *in-doubt* space was allocated should fail before

updating the quota system about its actual usage, this space might be lost. Should the amount of space in doubt approach a significant percentage of the quota, run the **mmcheckquota** command to account for the lost space.

To collect and display updated quota information about a group named `blueteam`, specify the **-g** and **-e** options:

```
mmquota -g blueteam -e
```

The system displays information similar to:

		Block Limits					File Limits				
Filesystem type		KB	quota	limit	in_doubt	grace	files	quota	limit	in_doubt	grace
Disk quotas for group blueteam (gid 100):											
fs1	GRP	45730	52000	99000	1335	none	411	580	990	19	none

See the “mmquota Command” on page 232 for complete usage information.

Activating quota limit checking

Quota limit checking can be activated for users, groups, filesets, or any combination of these three.

You can have quotas activated automatically whenever the file system is mounted by specifying the quota option (**-Q yes**) when creating (**mmcrfs -Q yes**) or changing (**mmchfs -Q yes**) a GPFS file system. When creating a file system, the default is to **not** have quotas activated, so you must specify this option if you want quotas activated.

The **mmquotaon** command is used to turn quota limit checking back on if it had been deactivated by issuing the **mmquotaoff** command. Specify the file system name, and whether user, group, or fileset quotas are to be activated. If you want all three - user, group, and fileset quotas activated, specify only the file system name. After quotas have been turned back on, issue the **mmcheckquota** command to count inode and space usage.

For example, to activate user quotas on the file system **fs1**, enter:

```
mmquotaon -u fs1
```

To confirm the change, enter:

```
mmfs fs1 -Q
```

The system displays output similar to:

flag	value	description
-Q	user	Quotas enforced

See the “mmquotaon Command” on page 251 and the “mmfs Command” on page 219 for complete usage information.

Deactivating quota limit checking

During normal operation, there is no need to deactivate quota enforcement. The only reason you might have to deactivate quota enforcement is when users are denied allocation that their quotas should allow, due to loss of quota information during node failure.

If this occurs, use the **mmcheckquota** command after reactivating quotas to reconcile allocation data. When quota enforcement is deactivated, disk space and file allocations are made without regard to limits.

The **mmquotaoff** command is used to deactivate quota limit checking. Specify the file system name and whether user, group, or fileset quotas, or any combination of these three, are to be deactivated. If you want all types of quotas deactivated, specify only the file system name.

For example, to deactivate only user quotas on the file system **fs1**, enter:

```
mmquotaoff -u fs1
```

To confirm the change, enter:

```
mmfsfs fs1 -Q
```

The system displays output similar to:

flag	value	description
-Q	group;fileset	Quotas enforced

See the “mmquotaoff Command” on page 249 and the “mmfsfs Command” on page 219 for complete usage information.

Creating file system quota reports

You can have GPFS prepare a quota report for a file system using the **mmrepquota** command.

The quota report lists:

1. Number of files used
2. Amount of disk space used
3. Current quota limits
4. In doubt quotas (disk space allocated but currently unaccounted for)
5. Grace period allowance to exceed the soft limit
6. Whether the quotas have been explicitly set (**e**), are default values (**d**), or initial values (**i**). The entry type also indicates whether or not default quotas are enabled for the file system (**default on** or **default off**).

GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmfsquota** command and the **mmrepquota** command and the values used to determine if quota limits have been exceeded will be double the value reported by the **ls** command.

Specify whether you want to list only user quota information (**-u** flag), group quota information (**-g** flag), or fileset quota information (**-j** flag) in the **mmrepquota** command. The default is to summarize all three quotas. If the **-e** flag is not specified, there is the potential to display negative usage values as the quota server may process a combination of up-to-date and back-level information. When issuing the **mmrepquota** command on a mounted file system, negative *in doubt* values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

To list the group quotas (**-g** option) for all file systems in the cluster (**-a** option), and print a report with header lines (**-v** option), enter:

```
mmrepquota -g -v -a
```

The system displays information similar to:

*** Report for GRP quotas on fs1

Block Limits							File Limits					entryType
Name	type	KB	quota	limit	in_doubt	grace	files	quota	limit	in_doubt	grace	
system	GRP	25088	0	0	209120	none	32	0	0	1078	none	default on
usr	GRP	435256	0	0	199712	none	11	0	0	899	none	d

See the “mmrepquota Command” on page 259 for complete usage information.

Restoring quota files

User, group, and fileset quota files can be restored from a backup copy of the original quota file. When restoring quota files, the backup file must be in the root directory of the file system. If a backup copy does not exist, an empty file will be created when the **mmcheckquota** command is issued.

The user, group, or fileset files can be restored from backup copies by issuing the **mmcheckquota** command with the appropriate options.

1. To restore the user quota file for the file system **fs1** from the backup file **userQuotaInfo**, enter:
`mmcheckquota -u userQuotaInfo fs1`
2. This will restore the user quota limits set for the file system, but the usage information will not be current. To bring the usage information to current values, the command must be reissued:
`mmcheckquota fs1`

If no backup files are available and the quota files are to be restored using a new file, these steps must be followed:

1. The existing corrupted quota files need to be removed:
 - a. Unmount the file system.
 - b. Disable quota management:
`mmchfs fs1 -Q no`
 - c. Remount the file system.
 - d. Remove the **user.quota**, **group.quota**, and **fileset.quota** files.
2. Enable quota management:
 - a. Unmount the file system.
 - b. Issue the following command:
`mmchfs fs1 -Q yes`
 - c. Remount the file system.
3. Reestablish quota limits by issuing the **mmedquota** command or the **mmdefedquota** command.
4. Gather the current quota usage values by issuing the **mmcheckquota** command.

Chapter 6. Managing GPFS access control lists and NFS export

Access control protects directories and files by providing a means of specifying who is granted access. GPFS access control lists are either traditional ACLs based on the POSIX model, or NFS V4 ACLs. NFS V4 ACLs are very different than traditional ACLs, and provide much more fine control of file and directory access. A GPFS file system can also be exported using NFS.

Management of GPFS access control lists (ACLs) and NFS export includes these topics:

- “Traditional GPFS ACL administration”
- “NFS V4 ACL administration” on page 52
- “NFS and GPFS” on page 57

Traditional GPFS ACL administration

Support for NFS V4 access control lists (ACLs) has been added to traditional ACL support. NFS V4 ACLs are very different than the traditional ones.

If you are using NFS V4 ACLs, see “NFS V4 ACL administration” on page 52. Both ACL types may coexist in a single GPFS file system.

Traditional GPFS ACLs are based on the POSIX model. Traditional GPFS access control lists (ACLs) extend the base permissions, or standard file access modes, of read (r), write (w), and execute (x) beyond the three categories of file owner, file group, and other users, to allow the definition of additional users and user groups. In addition, GPFS introduces a fourth access mode, control (c), which can be used to govern who can manage the ACL itself.

In this way, a traditional ACL can be created that looks like this:

```
#owner:jesmith
#group:team_A
user::rwx
group::rwx-
other::--x-
mask::rwx
user:alpha:r-x
group:audit:r-x-
group:system:rwx-
```

In this ACL:

- The first two lines are comments showing the file’s owner, **jesmith**, and group name, **team_A**
- The next three lines contain the base permissions for the file. These three entries are the minimum necessary for a GPFS ACL:
 1. The permissions set for the file owner (**user**), **jesmith**
 2. The permissions set for the owner’s **group**, **team_A**
 3. The permissions set for **other** groups or users outside the owner’s group and not belonging to any named entry
- The next line, with an entry type of **mask**, contains the maximum permissions allowed for any entries other than the owner (the **user** entry) and those covered by **other** in the ACL.
- The last three lines contain additional entries for specific users and groups. These permissions are limited by those specified in the mask entry, but you may specify any number of additional entries up to a memory page (approximately 4 K) in size.

Traditional GPFS ACLs are fully compatible with the base operating system permission set. Any change to the base permissions, using the **chmod** command, for example, modifies the corresponding GPFS ACL as well. Similarly, any change to the GPFS ACL is reflected in the output of commands such as **ls -l**. Note that the control (c) permission is GPFS specific. There is no comparable support in the base operating system commands. As a result, the (c) permission is visible only with the GPFS ACL commands.

Each GPFS file or directory has an *access ACL* that determines its access privileges. These ACLs control who is allowed to read or write at the file or directory level, as well as who is allowed to change the ACL itself.

In addition to an *access ACL*, a directory may also have a *default ACL*. If present, the default ACL is used as a base for the access ACL of every object created in that directory. This allows a user to protect all files in a directory without explicitly setting an ACL for each one.

When a new object is created, and the parent directory has a default ACL, the entries of the default ACL are copied to the new object's access ACL. After that, the base permissions for user, mask (or group if mask is not defined), and other, are changed to their intersection with the corresponding permissions from the mode parameter in the function that creates the object.

If the new object is a directory, its default ACL is set to the default ACL of the parent directory. If the parent directory does not have a default ACL, the initial access ACL of newly created objects consists only of the three required entries (user, group, other). The values of these entries are based on the mode parameter in the function that creates the object and the umask currently in effect for the process.

Administrative tasks associated with traditional GPFS ACLs are:

1. "Setting traditional GPFS access control lists"
2. "Displaying traditional GPFS access control lists" on page 51
3. "Changing traditional GPFS access control lists" on page 52
4. "Deleting traditional GPFS access control lists" on page 52

Setting traditional GPFS access control lists

Use the **mmputacl** command to set the access ACL of a file or subdirectory, or the default ACL of a directory. For example, to set the ACL for a file named **project2.history**, we can create a file named **project2.acl** that contains this:

```
user::rwx-
group::rwx-
other::--x-
mask::rwx-
user:alpha:r-xc
group:audit:rw--
group:system:rwx-
```

In the **project2.acl** file above,

- The first three lines are the required ACL entries setting permissions for the file's owner, the owner's group, and for processes that are not covered by any other ACL entry.
- The last three lines contain named entries for specific users and groups.
- Because the ACL contains named entries for specific users and groups, the fourth line contains the required mask entry, which is applied to all named entries (entries other than the **user** and **other**).

Once you are satisfied that the correct permissions are set in the ACL file, you can apply them to the target file with the **mmputacl** command. For example, to set permissions contained in the file **project2.acl** for the file **project2.history**, enter:

```
mmputacl -i project2.acl project2.history
```

To confirm the changes, enter:

```
mmgetacl project2.history
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwx
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwx #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

Although you can issue the **mmputacl** command without using the **-i** option to specify an ACL input file, and make ACL entries through standard input, you will probably find the **-i** option more useful for avoiding errors when creating a new ACL.

See the “mmputacl Command” on page 246 and the “mmgetacl Command” on page 196 for complete usage information.

Displaying traditional GPFS access control lists

Use the **mmgetacl** command to display the access ACL of a file or subdirectory, or the default ACL of a directory. For example, to display the ACL for the file **project2.history**, enter:

```
mmgetacl project2.history
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwx
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwx #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

The first two lines are comments displayed by the **mmgetacl** command, showing the owner and owning group. All entries containing permissions that are not allowed (because they are not set in the mask entry) display with a comment showing their effective permissions.

See the “mmgetacl Command” on page 196 for complete usage information.

Applying an existing traditional GPFS access control list

To apply the same traditional ACLs from one file or directory to another:

1. Issue the **mmgetacl** command with the **-o** option to place the information in an output file.
2. Apply the ACLs to the new file or directory by issuing the **mmputacl** command with the **-i** option.

For example, use the **-o** option to specify a file to which the ACL is written:

```
mmgetacl -o old.acl project2.history
```

Then, to assign the same permissions to another file, **project.notes**, enter:

```
mmputacl -i old.acl project.notes
```

To confirm the changes, enter:

```
mmgetacl project.notes
```

The information sent to standard output is similar to:

```
#owner:guest
#group:usr
user::rwx
group::rwx- #effective:rw--
other::--x-
mask::rw-c
user:alpha:rwx #effective:rw-c
group:audit:rwx- #effective:rw--
group:system:-w--
```

See the “mmgetacl Command” on page 196 and the “mmputacl Command” on page 246 for complete usage information.

Changing traditional GPFS access control lists

Use the **mmeditACL** command to change or create the traditional ACL of a file or directory, or the default ACL of a directory. For example, to interactively edit the ACL for the file **project2.history**, enter:

```
mmeditACL project2.history
```

The current ACL entries are displayed using the default editor, provided that the EDITOR environment variable specifies a complete path name. When the file is saved, the system displays information similar to:

```
mmeditACL: 6027-967 Should the modified ACL be applied? (yes) or (no)
```

After responding **yes**, the ACLs are applied.

See the “mmeditACL Command” on page 179 for complete usage information.

Deleting traditional GPFS access control lists

Use the **mmdeACL** command to delete the extended entries in a traditional ACL of a file or directory, or the default ACL of a directory. For example, to delete the ACL for the directory **project2**, enter:

```
mmdeACL project2
```

To confirm the deletion, enter:

```
mmgetacl project2
```

The system displays information similar to:

```
#owner:uno
#group:system
user::rwx
group::r-x-
other::--x-
```

You cannot delete the base permissions. These remain in effect after this command is executed.

See the “mmdeACL Command” on page 160 and the “mmgetacl Command” on page 196 for complete usage information.

NFS V4 ACL administration

AIX does not allow a file system to be NFS V4 exported unless it supports NFS V4 ACLs.

Normally, GPFS has POSIX ACLs (the default for the **mmcrfs** command is **-k posix**) so the file system must first be changed to allow NFS V4 ACLs (either exclusively using the **-k nfs4** flag, or along with POSIX, using the **-k all** flag). Once the file system is configured to allow NFS V4 ACLs, it may be exported and NFS V4 clients may mount and use the file system (including setting ACLs which will, of course, be NFS V4 style).

Depending on the value (**posix** | **nfs4** | **all**) of the **-k** parameter, one or both ACL types can be allowed for a given file system. Since ACLs are assigned on a per-file basis, this means that within the same file system one file may have an NFS V4 ACL, while another has a POSIX ACL. The type of ACL can be changed by using the **mmputacl** or **mmeditACL** command to assign a new ACL or by the **mmdeletACL** command (causing the permissions to revert to the mode which is in effect a POSIX ACL). At any point in time, only a single ACL can be associated with a file. Access evaluation is done as required by the ACL type associated with the file.

NFS V4 ACLs are represented in a completely different format than traditional ACLs. For detailed information on NFS V4 and its ACLs, refer to the paper, *NFS Version 4 Protocol* and other information found at: www.nfsv4.org.

In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual ACL entries can be flagged as being *inherited* (either by files, directories, both, or neither). Consequently, specifying the **-d** flag on the **mmputacl** command for an NFS V4 ACL is an error.

NFS V4 ACL Syntax

An NFS V4 ACL consists of a list of ACL entries. Where traditional ACLs can display one entry per line, the GPFS representation of NFS V4 ACL entries are three lines each, due to the increased number of available permissions beyond the traditional **rwxc**.

The first line has several parts separated by colons (':').

- The first part identifies the user or group.
- The second part displays a **rwxc** translation of the permissions that appear on the subsequent two lines.
- The third part is the ACL type. NFS V4 provides both an *allow* and *deny* type.

allow Means to allow (or permit) those permissions that have been selected with an 'X'.

deny Means to not allow (or deny) those permissions that have been selected with an 'X'.

- The fourth and final part is a list of flags indicating *inheritance*.

Valid flag values are:

FileInherit

Indicates that the ACL entry should be included in the initial ACL for files created in this directory.

DirInherit

Indicates that the ACL entry should be included in the initial ACL for subdirectories created in this directory (as well as the current directory).

InheritOnly

Indicates that the current ACL entry should *NOT* apply to the directory, but *SHOULD* be included in the initial ACL for objects created in this directory.

As in traditional ACLs, users and groups are identified by specifying the type and name. For example, **group:staff** or **user:bin**. NFS V4 provides for a set of special names that are not associated with a specific local UID or GID. These special names are identified with the keyword **special** followed by the NFS V4 name. These names are recognized by the fact that they end in with the character '@'. For example, **special:owner@** refers to the owner of the file, **special:group@** the owning group, and **special:everyone@** applies to all users.

The next two lines provide a list of the available access permissions that may be *allowed* or *denied*, based on the ACL type specified on the first line. A permission is selected using an 'X'. Permissions that are not specified by the entry should be left marked with '-' (minus sign).

These are examples of NFS V4 ACLs.

1. An ACL entry that explicitly allows **READ**, **EXECUTE** and **READ_ATTR** to the **staff** group on a file is similar to this:

```
group:staff:r-x-:allow
(X)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (X)READ_ATTR (-)READ_NAMED
(-)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

2. A Directory ACL is similar to this. It may include *inherit* ACL entries that do not apply to the directory itself, but instead become the initial ACL for any objects created within the directory.

```
special:group@:----:deny:DirInherit:InheritOnly
(X)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (X)READ_ATTR (-)READ_NAMED
(-)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

3. A complete NFS V4 ACL is similar to this:

```
#NFSv4 ACL
#owner:smithj
#group:staff
special:owner@:rwx::allow:FileInherit
(X)READ/LIST (X)WRITE/CREATE (X)MKDIR (-)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (-)READ_NAMED
(X)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED

special:owner@:rwx::allow:DirInherit:InheritOnly
(X)READ/LIST (X)WRITE/CREATE (X)MKDIR (-)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (-)READ_NAMED
(X)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED

user:smithj:rwx::allow
(X)READ/LIST (X)WRITE/CREATE (X)MKDIR (-)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (-)READ_NAMED
(X)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

ACL entries DELETE and DELETE_CHILD

The ACL entries **DELETE** and **DELETE_CHILD** require special considerations. The effect of various combinations of the **DELETE** attribute for a file, and the **DELETE_CHILD** attribute for its parent directory, is given in Table 3.

In this table, the columns refer to the ACL entry for a given file, and the rows refer to the ACL entry for its parent directory. The various combinations of these attributes produce one of these results:

Permit

Indicates that GPFS permits removal of a file with the combination of file and parent directory ACL entries specified.

Deny Indicates that GPFS denies (does not permit) removal of a file with the combination of file and parent directory ACL entries specified.

Removal of a file includes renaming the file, moving the file from one directory to another even if the file name remains the same, and deleting it.

Table 3. Removal of a file with ACL entries DELETE and DELETE_CHILD

	ACL Allows DELETE	ACL Denies DELETE	DELETE not specified	UNIX mode bits only
ACL Allows DELETE_CHILD	Permit	Permit	Permit	Permit
ACL Denies DELETE_CHILD	Permit	Deny	Deny	Deny
DELETE_CHILD not specified	Permit	Deny	Deny	Deny
UNIX mode bits only - wx permissions allowed	Permit	Permit	Permit	Permit

Table 3. Removal of a file with ACL entries DELETE and DELETE_CHILD (continued)

	ACL Allows DELETE	ACL Denies DELETE	DELETE not specified	UNIX mode bits only
UNIX mode bits only - no w or no x permissions allowed	Permit	Deny	Deny	Deny

The UNIX mode bits are used in cases where the ACL is not an NFS V4 ACL.

NFS V4 ACL translation

NFS V4 access requires that an NFS V4 ACL be returned to clients whenever the ACL is read. This means that if a traditional GPFS ACL is associated with the file, a translation to NFS V4 ACL format must be performed when the ACL is read by an NFS V4 client. Since this translation has to be done, an option (**-k nfs4**) is provided on the **mmgetacl** and **mmeditacl** commands, so that this translation can be seen locally as well.

It can also be the case that NFS V4 ACLs have been set for some file system objects (directories and individual files) prior to administrator action to revert back to a POSIX-only configuration. Since the NFS V4 access evaluation will no longer be performed, it is desirable for the **mmgetacl** command to return an ACL representative of the evaluation that will now occur (translating NFS V4 ACLs into traditional POSIX style). The **-k posix** option returns the result of this translation.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, the **mmgetacl** command returns the ACL in a format consistent with the file system setting:
 - If **posix** only, it is shown as a traditional ACL.
 - If **nfs4** only, it is shown as an NFS V4 ACL.
 - If **all** formats are supported, the ACL is returned in its true form.
2. The command **mmgetacl -k nfs4** always produces an NFS V4 ACL.
3. The command **mmgetacl -k posix** always produces a traditional ACL.
4. The command **mmgetacl -k native** always shows the ACL in its true form, regardless of the file system setting.

In general, users should continue to use the **mmgetacl** and **mmeditacl** commands without the **-k** flag, allowing the ACL to be presented in a form appropriate for the file system setting. Since the NFS V4 ACLs are more complicated and therefore harder to construct initially, users that want to assign an NFS V4 ACL should use the command **mmeditacl -k nfs4** to start with a translation of the current ACL, and then make any necessary modifications to the NFS V4 ACL that is returned.

Setting NFS V4 access control lists

There is no option on the **mmputacl** command to identify the type (traditional or NFS V4) of ACL that is to be assigned to a file. Instead, the ACL is assumed to be in the traditional format unless the first line of the ACL is:

```
#NFSv4 ACL
```

The lines that follow the first one are then processed according to the rules of the expected ACL type.

An NFS V4 ACL is similar to this:

```
#NFSv4 ACL
#owner:root
#group:system
special:owner@:rwx:allow
```

```
(X)READ/LIST (X)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL (-)READ_ATTR (-)READ_NAMED
(X)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED
```

```
special:owner@:----:deny
```

```
(-)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (-)READ_ATTR (X)READ_NAMED
(-)DELETE (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (X)WRITE_NAMED
```

```
user:guest:r-xc:allow
```

```
(X)READ/LIST (-)WRITE/CREATE (-)MKDIR (X)SYNCHRONIZE (X)READ_ACL (-)READ_ATTR (-)READ_NAMED
(X)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

```
user:guest:----:deny
```

```
(-)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (-)READ_ATTR (X)READ_NAMED
(-)DELETE (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED
```

This ACL shows four ACL entries (an allow and deny entry for each of **owner@** and **guest**).

In general, constructing NFS V4 ACLs is more complicated than traditional ACLs. Users new to NFS V4 ACLs may find it useful to start with a traditional ACL and allow either **mmgetacl** or **mmeditACL** to provide the NFS V4 translation, using the **-k nfs4** flag as a starting point when creating an ACL for a new file.

Displaying NFS V4 access control lists

The **mmgetacl** command displays an existing ACL regardless of its type (traditional or NFS V4). The format of the ACL that is returned depends on the file system setting (**-k** flag), as well as the format of the actual ACL associated with the file. For details, see “NFS V4 ACL translation” on page 55.

Applying an existing NFS V4 access control lists

This function is identical, whether using traditional or NFS V4 ACLs. See “Applying an existing traditional GPFS access control list” on page 51.

Changing NFS V4 access control lists

This function is identical, whether using traditional or NFS V4 ACLs. See “Changing traditional GPFS access control lists” on page 52.

Deleting NFS V4 access control lists

Use the **mmdelACL** command to delete NFS V4 ACLs. Once the ACL has been deleted, permissions revert to the mode bits. If the **mmgetacl** command is then used to display the ACL (**mmgetacl -k native**), it appears as a traditional GPFS ACL.

When assigning an ACL to a file that already has an NFS V4 ACL, there are some NFS rules that must be followed. Specifically, in the case of a directory, there will **not** be two separate (access and default) ACLs, as there are with traditional ACLs. NFS V4 requires a single ACL entity and allows individual ACL entries to be flagged if they are to be inherited. Consequently, **mmpuACL -d** is not allowed if the existing ACL was the NFS V4 type, since this attempts to change **only** the default ACL. Likewise **mmpuACL** (without the **-d** flag) is not allowed because it attempts to change only the access ACL, leaving the default unchanged. To change such an ACL, use the **mmeditACL** command to change the entire ACL as a unit. Alternatively, use the **mmdelACL** command to remove an NFS V4 ACL, followed by the **mmpuACL** command.

Considerations when using GPFS with NFS V4 ACLs

There are several constraints that you need to consider when using GPFS with NFS V4 ACLs. For a complete description of these restrictions, see “GPFS exceptions and limitations to NFS V4 ACLs” on page 394.

NFS and GPFS

GPFS file systems may be exported using the Network File System (NFS) protocol from one or more nodes. After export, normal access to the file system can proceed from GPFS cluster nodes or NFS client nodes.

I **Note:** GPFS on Windows does not provide NFS integration.

Considerations for the interoperability of a GPFS file system include:

- “Exporting a GPFS file system using NFS”
- “NFS usage of GPFS cache” on page 59
- “Synchronous writing using NFS” on page 59
- “Unmounting a file system after NFS export” on page 59
- “NIS automount considerations” on page 60
- “Clustered NFS and GPFS on Linux” on page 60

Exporting a GPFS file system using NFS

To export a GPFS file system:

1. Create and mount the GPFS file system. In the examples, we assume a file system with a local mount point of **/gpfs**.

For performance reasons, some NFS implementations cache file information on the client. Some of the information (for example, file state information such as file size and timestamp) is not kept up-to-date in this cache. The client may view stale inode data (on **ls -l**, for example) if exporting a GPFS file system with NFS.

If this is not acceptable for a given installation, caching can be turned off by mounting the file system on the client using the appropriate operating system mount option (for example, **-o noac** on Linux NFS clients). Turning off NFS caching results in extra file systems operations to GPFS, and negatively affect its performance.

2. Make sure that the clocks of all nodes in the GPFS cluster are synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations, may be disrupted.

NFS relies on metadata timestamps to validate the local operating system cache. If the same directory is either NFS-exported from more than one node, or is accessed with both the NFS and GPFS mount point, it is critical that clocks on all nodes that access the file system (GPFS nodes and NFS clients) are constantly synchronized using appropriate software (for example, NTP). Failure to do so may result in stale information seen on the NFS clients.

3. Ensure that NFS is properly configured and running.

For Linux nodes, information on configuring NFS can be obtained at www.linuxdoc.org.

For AIX nodes, information on configuring NFS can be obtained at publib.boulder.ibm.com/infocenter/pseries/index.jsp.

4. Edit the **/etc/exports** file. Include all of the file systems to be exported. For example, to export the directory **/gpfs/dir1** to the **cluster1** node, enter:

```
/gpfs/dir1 rw,access=cluster1
```

Export considerations

Keep these points in mind when exporting a GPFS file system to NFS. The operating system being used and the version of NFS might require special handling or consideration.

Linux export considerations: For Linux nodes only, issue the **exportfs -ra** command to initiate a reread of the **/etc/exports** file.

Starting with Linux kernel version 2.6, an **fsid** value must be specified for each GPFS file system that is exported on NFS. The format of the entry in **/etc/exports** for the GPFS directory **/gpfs/dir1** looks like this:

```
/gpfs/dir1 cluster1(rw,fsid=745)
```

The administrator must assign **fsid** values subject to the following conditions:

1. The values must be unique for each file system.
2. The values must not change after reboots. The file system should be unexported before any change is made to an already assigned **fsid**.
3. Entries in the **/etc/exports** file are not necessarily file system roots. You can export multiple directories within a file system. In the case of different directories of the same file system, the **fsid** should be the same. For example, in the GPFS file system **/gpfs**, if two directories are exported (**dir1** and **dir2**), the entries may look like this:

```
/gpfs/dir1 cluster1(rw,fsid=745)  
/gpfs/dir2 cluster1(rw,fsid=745)
```
4. If a GPFS file system is exported from multiple nodes, the **fsids** should be the same on all nodes.
5. If you want to export different directories of the same file system, use different **fsids**.

Large installations with hundreds of compute nodes and a few login nodes or NFS-exporting nodes require tuning of the GPFS parameters **maxFilesToCache** and **maxStatCache** with the **mmchconfig** command. The general suggestion is for the compute nodes to set **maxFilesToCache** to about 200. The login or NFS nodes should set this parameter much higher, with **maxFilesToCache** set to 1000 and **maxStatCache** set to 50000.

This tuning is required for the GPFS token manager (file locking), which can handle approximately 1,000,000 files in memory. The default value of **maxFilesToCache** is 1000 and the default value of **maxStatCache** is $4 * \text{maxFilesToCache}$, so that by default, each node holds 5000 tokens. The token manager must keep track of a total number of tokens, which equals $5000 * \text{number of nodes}$. This will exceed the memory limit of the token manager on large configurations.

GPFS does not support NFS V4 exporting GPFS file systems from Linux nodes. NFS V3 is acceptable.

If you are running at SLES 9 SP 1, the kernel defines the **sysctl** variable **fs.nfs.use_underlying_lock_ops**, which determines whether the NFS **lockd** is to consult the file system when granting advisory byte-range locks. For distributed file systems like GPFS, this must be set to **true** (the default is **false**).

You can query the current setting by issuing the command:

```
sysctl fs.nfs.use_underlying_lock_ops
```

Alternatively, the **fs.nfs.use_underlying_lock_ops = 1** record can be added to **/etc/sysctl.conf**. This record must be applied after initially booting the node, and after each reboot, by issuing the command:

```
sysctl -p
```

Because the **fs.nfs.use_underlying_lock_ops** variable is currently not available in SLES 9 SP 2 or later, when NFS-exporting a GPFS file system, ensure that your NFS server nodes are at the SP 1 level (unless this variable is made available in later service packs).

For additional considerations when NFS exporting your GPFS file system, refer to "File system creation considerations" in *GPFS: Concepts, Planning, and Installation Guide*.

AIX export considerations: AIX does not allow a file system to be exported by NFS V4 unless it supports NFS V4 ACLs.

NFS export considerations for versions prior to NFS V4: For NFS exported file systems, the version of NFS you are running with may have an impact on the number of inodes you need to cache, as set by both the `maxStatCache` and `maxFilesToCache` parameters on the `mmchconfig` command. The implementation of the `ls` command differs from NFS V2 to NFS V3. The performance of the `ls` command in NFS V3 in part depends on the caching ability of the underlying file system. Setting the cache large enough will prevent rereading inodes to complete an `ls` command, but will put more of a CPU load on the token manager.

Also, the clocks of all nodes in your GPFS cluster must be synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations, may be disrupted.

NFS V4 export considerations: For information on NFS V4, see the paper, *NFS Version 4 Protocol* and other information found at: www.nfsv4.org.

To export a GPFS file system using NFS V4, there are two file system settings that must be in effect. These attributes can be queried using the `mmlsfs` command, and set using the `mmcrfs` and `mmchfs` commands.

1. The `-D nfs4` flag is required. Conventional NFS access would not be blocked by concurrent file system reads or writes (this is the POSIX semantic). NFS V4 however, not only allows for its requests to block if conflicting activity is happening, it insists on it. Since this is an NFS V4 specific requirement, it must be set before exporting a file system.

flag value	description

<code>-D nfs4</code>	File locking semantics in effect

2. The `-k nfs4` or `-k all` flag is required. Initially, a file system has the `-k posix` setting, and only traditional GPFS ACLs are allowed. To export a file system using NFS V4, NFS V4 ACLs must be enabled. Since NFS V4 ACLs are vastly different and affect several characteristics of the file system objects (directories and individual files), they must be explicitly enabled. This is done either exclusively, by specifying `-k nfs4`, or by allowing `all` ACL types to be stored.

flag value	description

<code>-k all</code>	ACL semantics in effect

NFS usage of GPFS cache

Exporting a GPFS file system from a node may result in significant additional demands on the resources at that node. Depending on the number of NFS clients, their demands, and specific mount options, you may want to increase either one or both of the `maxFilesToCache` and `pagepool` configuration options. See the `mmchconfig` command.

You may also choose to restrict the use of the NFS server node through the normal GPFS path and not use it as either a file system manager node or an NSD server.

Synchronous writing using NFS

With Linux, `write` operations are usually asynchronous. If synchronous writes are required over NFS, edit the `/etc/exports` file to include `sync,no_wdelay`.

Unmounting a file system after NFS export

Because NFS use of a GPFS file system may result in a file being held, attempting to unmount a GPFS file system may return a 'Device is busy' error. If this occurs, stop the NFS daemons before attempting to unmount the file system at the NFS server. On Linux, issue this command:

```
/etc/rc.d/init.d/nfs stop
```

On AIX, issue this command:

```
stopsrc -g nfs
```

NFS can be restarted after the unmount completes. On Linux, issue this command:

```
/etc/rc.d/init.d/nfs start
```

On AIX, issue this command:

```
startsrc -g nfs
```

NIS automount considerations

The default file system type when using the automounter daemon is NFS. When the **-fstype** option is not specified, and the server is the local node, a soft-mount of the local directory is done at the desired mount point. JFS is assumed as the only handler of local directories. A GPFS file system local soft-mount does not work implicitly, since the mount request is passed to JFS which then produces an error. When specifying **-fstype mmfs** the local soft-mount works because the mount is then passed to GPFS instead of JFS.

A GPFS soft-mount does not automatically unmount. Setting **-fstype nfs3** causes the local server mounts to always go through NFS. This allows you to have the same **auto.map** file on all nodes whether the server is local or not, and the automatic unmount will occur. If you want local soft-mounts of GPFS file systems while other nodes perform NFS mounts, you should have different **auto.map** files on the different classes of nodes. This should improve performance on the GPFS nodes as they will not have to go through NFS.

Clustered NFS and GPFS on Linux

In addition to the traditional exporting of GPFS file systems using NFS, GPFS allows you to configure a subset of the nodes in the cluster to provide a highly available solution for exporting GPFS file systems via NFS.

The participating nodes are designated as Cluster NFS (CNFS) member nodes and the entire setup is frequently referred to as CNFS or CNFS cluster.

In this solution, all CNFS nodes export the same file systems to the NFS clients. When one of the CNFS nodes fails, the NFS serving load moves from the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover.

For more information about CNFS, see *General Parallel File System: Advanced Administration Guide*.

Chapter 7. Communicating file access patterns to GPFS

GPFS attempts to recognize the pattern of accesses that an application makes to an open file, and optimizes its behavior accordingly.

For example, GPFS can recognize sequential reads and therefore prefetch blocks in advance of when they are required by the application. However, in many cases GPFS does not recognize the access pattern of the application, or cannot optimize its data transfers. In these situations, performance may improve if the application explicitly discloses aspects of its access pattern to GPFS using the **gpfs_fcntl()** subroutine. These subroutines are exploited by MPI Version 2.0.

The **gpfs_fcntl()** subroutine allows application programs to pass two classes of file access information giving GPFS an opportunity to improve throughput and latency of file system requests:

1. Hints
2. Directives

Hints allow an application to disclose its future accesses to GPFS. Hints are always optional. Adding or removing hints from a program, even incorrectly specified hints, will never alter the meaning of a program. Hints can only affect the performance of an application. The hints that may be passed to GPFS are:

1. **gpfsAccessRange_t**
2. **gpfsClearFileCache_t**
3. **gpfsFreeRange_t**
4. **gpfsMultipleAccessRange_t**

Note: GPFS is free to silently ignore a hint if system resources do not permit the hint to be processed.

In contrast, directives are stronger than hints. They may affect program semantics and must be either carried out by GPFS or return an error. The directives which may be passed to GPFS are:

1. **gpfsCancelHints_t**
2. **gpfsDataShipMap_t**
3. **gpfsDataShipStart_t**
4. **gpfsDataShipStop_t**

To communicate hints and directives to GPFS, an application program builds a data structure in memory, then passes it to GPFS. This data structure consists of:

- A fixed length header, mapped by **gpfsFcntlHeader_t**.
- Followed by a variable number of, and any combination of, hints and directives.

Hints and directives may be mixed within a single **gpfs_fcntl()** subroutine, and are performed in the order that they appear. A subsequent hint or directive may cancel out a preceding one.

The header and hints and directives that follow it are defined as C structures.

The **gpfs_fcntl()** subroutine takes the handle of the opened file as its first parameter and the address of the data structure as its second parameter. For complete definitions of the **gpfs_fcntl()** subroutine, the header, hints, directives, and other operations, see “**gpfs_fcntl()** Subroutine” on page 298.

Chapter 8. GPFS commands

A list of all the GPFS commands and a short description of each is presented in this topic.

Table 4 summarizes the GPFS-specific commands.

Table 4. GPFS commands

Command	Purpose
"mmadddisk Command" on page 66	Adds disks to a GPFS file system.
"mmaddnode Command" on page 70	Adds nodes to a GPFS cluster.
"mmapplypolicy Command" on page 73	Deletes files or migrates file data between storage pools in accordance with policy rules.
"mmauth Command" on page 78	Manages secure access to GPFS file systems.
"mmbackup Command" on page 83	Backs up a GPFS file system to a backup server.
"mmchattr Command" on page 86	Changes attributes of one or more GPFS files.
"mmchcluster Command" on page 89	Changes GPFS cluster configuration data.
"mmchconfig Command" on page 92	Changes GPFS configuration parameters.
"mmchdisk Command" on page 100	Changes state or parameters of one or more disks in a GPFS file system.
"mmcheckquota Command" on page 104	Checks file system user, group, and fileset quotas.
"mmchfileset Command" on page 107	Changes the attributes of a GPFS fileset.
"mmchfs Command" on page 109	Changes the attributes of a GPFS file system.
"mmchmgr Command" on page 114	Assigns a file system manager node.
"mmchnode Command" on page 116	Changes node attributes.
"mmchnsd Command" on page 120	Changes NSD configuration parameters.
"mmchpolicy Command" on page 122	Establish policy rules for a GPFS file system.
"mmcrcluster Command" on page 124	Creates a GPFS cluster from a set of nodes.
"mmcrfileset Command" on page 128	Creates a GPFS fileset.
"mmcrfs Command" on page 130	Creates a GPFS file system.
"mmcrnsd Command" on page 137	Creates cluster-wide names for NSDs used by GPFS.
"mmcrsnapshot Command" on page 142	Creates a snapshot of an entire GPFS file system at a single point in time.
"mmcrvsd Command" on page 145	Creates virtual shared disks for use by GPFS.
"mmdefedquota Command" on page 150	Sets default quota limits for a file system.
"mmdefquotaoff Command" on page 152	Deactivates default quota limit usage for a file system.
"mmdefquotaon Command" on page 154	Activates default quota limit usage for a file system.
"mmdefragfs Command" on page 157	Reduces disk fragmentation by increasing the number of full free blocks available to the file system.
"mmdelacl Command" on page 160	Deletes a GPFS access control list.
"mmdeldisk Command" on page 162	Deletes disks from a GPFS file system.
"mmdelfileset Command" on page 165	Deletes a GPFS fileset.
"mmdelfs Command" on page 168	Removes a GPFS file system.
"mmdelnode Command" on page 170	Removes one or more nodes from a GPFS cluster.
"mmdelnsd Command" on page 172	Deletes NSDs from the GPFS cluster.

Table 4. GPFS commands (continued)

Command	Purpose
"mmdelsnapshot Command" on page 174	Deletes a GPFS snapshot.
"mmdf Command" on page 176	Queries available file space on a GPFS file system.
"mmeditACL Command" on page 179	Creates or changes a GPFS access control list.
"mmedquota Command" on page 182	Sets quota limits.
"mmexportfs Command" on page 185	Export a file system from the cluster.
"mmfsck Command" on page 187	Checks and repairs a GPFS file system.
"mmfsctl Command" on page 192	Issues a file system control request.
"mmgetacl Command" on page 196	Displays the GPFS access control list of a file or directory.
"mmgetstate Command" on page 199	Displays the state of the GPFS daemon on one or more nodes.
"mmimportfs Command" on page 202	Import a file system into the cluster.
"mmlinkfileset Command" on page 205	Creates a junction that references the root directory of a GPFS fileset.
"mmlsattr Command" on page 207	Queries file attributes.
"mmlscluster Command" on page 209	Displays the current configuration data for a GPFS cluster.
"mmlsconfig Command" on page 211	Displays the configuration data for a GPFS cluster.
"mmlsdisk Command" on page 213	Displays the current configuration and state of disks in a file system.
"mmlsfileset Command" on page 216	Displays status and attributes of GPFS filesets.
"mmlsfs Command" on page 219	Displays file system attributes.
"mmlsmgr Command" on page 223	Displays which node is the file system manager for the specified file systems.
"mmlsmount Command" on page 225	Lists the nodes that have a given GPFS file system mounted.
"mmlnsd Command" on page 227	Display NSD information for the GPFS cluster.
"mmlspolicy Command" on page 230	Displays policy information.
"mmlsquota Command" on page 232	Displays quota information for a user, group, or fileset.
"mmlssnapshot Command" on page 235	Displays GPFS snapshot information for the specified file system.
"mmm mount Command" on page 237	Mounts GPFS file systems on one or more nodes in the cluster.
"mmnsddiscover Command" on page 239	Rediscovered paths to the specified network shared disks.
"mmpmon Command" on page 241	Monitors GPFS performance on a per-node basis.
"mmputacl Command" on page 246	Sets the GPFS access control list for the specified file or directory.
"mmquotaoff Command" on page 249	Deactivates quota limit checking.
"mmquotaon Command" on page 251	Activates quota limit checking.
"mmremotefcluster Command" on page 253	Manages information about remote clusters.
"mmremotefs Command" on page 256	Manages the information needed for mounting remote GPFS file systems.
"mmrepquota Command" on page 259	Displays file system user, group, and fileset quotas.
"mmrestorefs Command" on page 262	Restores a file system from a GPFS snapshot.
"mmrestripefile Command" on page 265	Performs a repair operation over the specified list of files.

Table 4. GPFS commands (continued)

Command	Purpose
“mmrestripefs Command” on page 268	Rebalances or restores the replication factor of all files in a file system.
“mmrpldisk Command” on page 271	Replaces the specified disk.
“mmshutdown Command” on page 275	Unmounts all GPFS file systems and stops GPFS on one or more nodes.
“mmsnapdir Command” on page 277	Creates and deletes invisible directories that connect to the snapshots of a GPFS file system, and changes the name of the snapshots subdirectory.
“mmstartup Command” on page 280	Starts the GPFS subsystem on one or more nodes.
“mmtracectl Command” on page 282	Sets up and enables GPFS tracing.
“mmumount Command” on page 285	Unmounts GPFS file systems on one or more nodes in the cluster.
“mmunlinkfileset Command” on page 288	Removes the junction to a GPFS fileset.

mmadddisk Command

Adds disks to a GPFS file system.

Synopsis

mmadddisk *Device* {" *DiskDesc*;*DiskDesc*..." | **-F** *DescFile*} [-**a**] [-**r**] [-**v** {yes | **no**}] [-**N** {*Node*;*Node*...} | *NodeFile* | *NodeClass*}]

Description

Use the **mmadddisk** command to add disks to a GPFS file system. This command optionally rebalances an existing file system after adding disks when the **-r** flag is specified. The **mmadddisk** command does not require the file system to be unmounted before issuing the command. The file system can be in use while the command is run.

Device must be the first parameter.

The **-N** parameter can be used only in conjunction with the **-r** option.

To add disks to a GPFS file system, you first must decide if you will:

1. Create new disks using the **mmcrnsd** command.

You should also decide whether to use the rewritten disk descriptor file produced by the **mmcrnsd** command, or create a new list of disk descriptors. When using the rewritten file, the *Disk Usage* and *Failure Group* specifications will remain the same as specified on the **mmcrnsd** command.

2. Select disks no longer in use in any file system. Issue the **mmlnsd -F** command to display the available disks.

Parameters

Device

The device name of the file system to which the disks are added. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

DiskDesc

A descriptor for each disk to be added. Each descriptor is delimited by a semicolon (;) and the entire list must be enclosed in quotation marks (' or ").

The current maximum number of disk descriptors that can be defined for any single file system is 268 million. However, to achieve this maximum limit, you must recompile GPFS. The actual number of disks in your file system may be constrained by products other than GPFS that you have installed. Refer to the individual product documentation.

A disk descriptor is defined as (second, third and sixth fields reserved):

DiskName:::DiskUsage:FailureGroup:::StoragePool

DiskName

You must specify the name of the NSD previously created by the **mmcrnsd** command. For a list of available disks, issue the **mmlnsd -F** command.

DiskUsage

Specify a disk usage or accept the default:

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly

Indicates that the disk contains data and does not contain metadata.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

FailureGroup

A number identifying the failure group to which this disk belongs. You can specify any value from -1 to 4000 (where -1 indicates that the disk has no point of failure in common with any other disk). If you do not specify a failure group, the value defaults to the node number of the first NSD server defined in the NSD server list plus 4000. If you do not specify an NSD server list, the value defaults to -1. GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter should be placed in the same failure group.

If replication of **-m** or **-r** is set to 2 for the file system, storage pools must have two failure groups for the commands to work properly.

StoragePool

Specifies the storage pool to which the disk is to be assigned. If this name is not provided, the default is **system**.

Only the system storage pool can contain **metadataOnly**, **dataAndMetadata**, or **descOnly** disks. Disks in other storage pools must be **dataOnly**.

-F DescFile

Specifies a file containing a list of disk descriptors, one per line. You may use the rewritten *DiskDesc* file created by the **mmcrnsd** command or create your own file. When using the *DiskDesc* file created by the **mmcrnsd** command, the values supplied on input to the command for *Disk Usage* and *FailureGroup* are used. When creating your own file, you must specify these values or accept the system defaults.

-N { Node[,Node...] | NodeFile | NodeClass }

Specifies the nodes that are to participate in the restripe of the file system after the specified disks have been made available for use by GPFS. This parameter can be used only in conjunction with the **-r** option. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the restripe of the file system).

For general information on how to specify node names, see "Specifying nodes as input to GPFS commands" on page 2.

Options

- a** Specifies asynchronous processing. If this flag is specified, the **mmadddisk** command returns after the file system descriptor is updated and the rebalancing scan is started; it does not wait for rebalancing to finish. If no rebalancing is requested (the **-r** flag not specified), this option has no effect.
- r** Rebalance all existing files in the file system to make use of new disks.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

-v {yes | no}

Verify that specified disks do not belong to an existing file system. The default is **-v yes**. Specify **-v no** only when you want to reuse disks that are no longer needed for an existing file system. If the command is interrupted for any reason, you must use the **-v no** option on the next invocation of the command.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmadddisk** command.

You may issue the **mmadddisk** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To add the disks whose descriptors are located in **./disk_pools** to the file system **fs1** and rebalance the existing files after it is added, issue this command:

```
mmadddisk fs1 -F ./disk_pools -r
```

The system displays information similar to:

```
GPFS: 6027-531 The following disks of fs1 will be formatted on node
k148n07.kgn.ibm.com:
  gpfs10nsd: size 2202180 KB
Extending Allocation Map
Creating Allocation Map for storage pool 'pool2'
  75 % complete on Thu Feb 16 13:57:52 2006
 100 % complete on Thu Feb 16 13:57:54 2006
Flushing Allocation Map for storage pool 'pool2'
GPFS: 6027-535 Disks up to size 24 GB can be added to storage
pool pool2.
Checking allocation map for storage pool system
  62 % complete on Thu Feb 16 13:58:03 2006
 100 % complete on Thu Feb 16 13:58:06 2006
Checking allocation map for storage pool pool1
  62 % complete on Thu Feb 16 13:58:11 2006
 100 % complete on Thu Feb 16 13:58:14 2006
Checking allocation map for storage pool pool2
  63 % complete on Thu Feb 16 13:58:19 2006
 100 % complete on Thu Feb 16 13:58:22 2006
GPFS: 6027-1503 Completed adding disks to file system fs1.
mmadddisk: 6027-1371 Propagating the cluster configuration
data to all affected nodes. This is an asynchronous process.
Restripping fs1 ...
Thu Feb 16 13:58:48 est 2006: mmcommon pushSdr_async: mmsdrfs
propagation started
Thu Feb 16 13:58:52 est 2006: mmcommon pushSdr_async: mmsdrfs
```

```
propagation completed; mmdsh rc = 0
GPFS: 6027-589 Scanning file system metadata, phase 1 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 2 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 3 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-589 Scanning file system metadata, phase 4 ...
GPFS: 6027-552 Scan completed successfully.
GPFS: 6027-565 Scanning user file metadata ...
  68 % complete on Thu Feb 16 13:59:06 2006
 100 % complete on Thu Feb 16 13:59:07 2006
GPFS: 6027-552 Scan completed successfully.
Done
```

See also

“mmchdisk Command” on page 100

“mmcrnsd Command” on page 137

“mmdeldisk Command” on page 162

“mmlsdisk Command” on page 213

“mmlsnsd Command” on page 227

Location

/usr/lpp/mmfs/bin

mmaddnode Command

Adds nodes to a GPFS cluster.

Synopsis

mmaddnode -N {*NodeDesc[,NodeDesc...]* | *NodeFile*}

Description

Use the **mmaddnode** command to add nodes to an existing GPFS cluster. On each new node, a mount point directory and character mode device is created for each GPFS file system.

You must follow these rules when adding nodes to a GPFS cluster:

- You may issue the command only from a node that already belongs to the GPFS cluster.
- While a node may mount file systems from multiple clusters, the node itself may only be added to a single cluster using the **mmcrcluster** or **mmaddnode** command.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command to add those nodes.

Parameters

-N *NodeDesc[,NodeDesc...] | NodeFile*

Specifies node descriptors, which provide information about nodes to be added to the cluster.

NodeFile

Specifies a file containing a list of node descriptors (see below), one per line, to be added to the cluster.

NodeDesc[,NodeDesc...]

Specifies the list of nodes and node designations to be added to the GPFS cluster. Node descriptors are defined as:

NodeName:NodeDesignations:AdminNodeName

where:

NodeName

Specifies the host name or IP address of the node for GPFS daemon-to-daemon communication.

The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Aliased interfaces are not allowed. Use the original address or a name that is resolved by the **host** command to that original address. You can specify a node using any of these forms:

- Short host name (for example, h135n01)
- Long, fully-qualified, host name (for example, h135n01.ibm.com)
- IP address (for example, 7.111.12.102)

Regardless of which form you use, GPFS will resolve the input to a host name and an IP address and will store these in its configuration files. It is expected that those values will not change while the node belongs to the cluster.

NodeDesignations

An optional, "-" separated list of node roles:

- **manager** | **client** – Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is **client**. Windows nodes cannot be designated as manager nodes.
- **quorum** | **nonquorum** – Indicates whether a node is counted as a quorum node. The default is **nonquorum**. Windows nodes cannot be designated as quorum nodes.

AdminNodeName
Specifies an optional field that consists of a node name to be used by the administration commands to communicate between nodes. If *AdminNodeName* is not specified, the *NodeName* value is used.

You must provide a *NodeDesc* for each node to be added to the GPFS cluster.

Options

NONE

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmaddnode** command.

You may issue the **mmaddnode** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To add nodes **k164n06** and **k164n07** as quorum nodes, designating **k164n06** to be available as a **manager** node, issue this command:

```
mmaddnode -N k164n06:quorum-manager,k164n07:quorum
```

To confirm the addition, issue this command:

```
mmfsccluster
```

The system displays information similar to:

```
GPFS cluster information
=====
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:    /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

GPFS cluster configuration servers:

Primary server: k164n07.kgn.ibm.com
Secondary server: k164n04.kgn.ibm.com

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n07.kgn.ibm.com	198.117.68.71	k164n07.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164n06.kgn.ibm.com	quorum-manager

See also

“mmchconfig Command” on page 92

“mmcrcluster Command” on page 124

“mmchcluster Command” on page 89

“mmdelnode Command” on page 170

“mmlscluster Command” on page 209

Location

/usr/lpp/mmfs/bin

mmapplypolicy Command

Deletes files or migrates file data between storage pools in accordance with policy rules.

Synopsis

```
mmapplypolicy {Device | Directory} [-I {yes | defer | test}] [-N {Node[,Node...] | NodeFile | NodeClass}]  
[-P PolicyFile] [-D yyyy-mm-dd [ @hh:mm[:ss]]] [-L n] [-M Name=Value...] [-S snapshotName] [-s  
WorkDirectory]
```

Description

Use the **mmapplypolicy** command to manage the placement and replication of data within GPFS storage pools. It can also be used to delete files from GPFS. You may issue the **mmapplypolicy** command from any node in the GPFS cluster that has the file system mounted.

Any given file is a potential candidate for at most one **MIGRATE** or **DELETE** operation during one invocation of the **mmapplypolicy** command.

A file that matches an **EXCLUDE** rule is not subject to any subsequent **MIGRATE** or **DELETE** rules. You should carefully consider the order of rules within a policy to avoid unintended consequences.

For detailed information on GPFS policies, see the chapter *Policy-based data management for GPFS in General Parallel File System: Advanced Administration Guide*.

| This command cannot be run from a Windows node.

Parameters

Device

Specifies the device name of the file system from which files are to be deleted or migrated. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**. If specified, this must be the first parameter.

Directory

Specifies the fully-qualified path name of a GPFS file system subtree from which files are to be deleted or migrated. If specified, this must be the first parameter.

-I {**yes** | **defer** | **test**}

Specifies what actions the **mmapplypolicy** command performs on files:

- yes** Indicates that all applicable **MIGRATE** and **DELETE** policy rules are run, and the data movement between pools is done during the processing of the **mmapplypolicy** command. This is the default action.
- defer** Indicates that all applicable **MIGRATE** and **DELETE** policy rules are run, but actual data movement between pools is deferred until the next **mmrestripefs** or **mmrestripefile** command.
- test** Indicates that all policy rules are evaluated, but the **mmapplypolicy** command only displays the actions that would be performed had **-I defer** or **-I yes** been specified. There is no actual deletion of files or data movement between pools. This option is intended for testing the effects of particular policy rules.

-N { *Node*[,*Node*...] | *NodeFile* | *NodeClass*}

Specifies the list of nodes that will run parallel instances of policy code in the GPFS home cluster. This command supports all defined node classes. The default is to run on the node where the **mmapplypolicy** command is running.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

-P PolicyFile

Specifies the name of the file containing the policy rules to be applied. If not specified, the policy rules currently in effect for the file system are used.

Options

-D yyyy-mm-dd[@hh:mm[:ss]]

Specifies a date and optionally a (UTC) time as *year-month-day* at *hour:minute:second*.

The **mmapplypolicy** command evaluates policy rules as if it were running on the date and time specified by the **-D** flag. This can be useful for planning or testing policies, to see how the **mmapplypolicy** command would act in the future. If this flag is omitted, the **mmapplypolicy** command uses the current date and (UTC) time. If a date is specified but not a time, the time is assumed to be 00:00:00.

-L n Controls the level of information displayed by the **mmapplypolicy** command. Larger values indicate the display of more detailed information. These terms are used:

candidate file

A file that matches a **MIGRATE** or **DELETE** policy rule.

chosen file

A candidate file that has been scheduled for migration or deletion.

These are the valid values for *n*:

- 0** Displays only serious errors.
- 1** Displays some information as the command runs, but not for each file. This is the default.
- 2** Displays each chosen file and the scheduled migration or deletion action.
- 3** All of the above, plus displays each candidate file and the applicable rule.
- 4** All of the above, plus displays each explicitly **EXCLUDE**'ed file, and the applicable rule.
- 5** All of the above, plus displays the attributes of candidate and **EXCLUDE**'ed files.
- 6** All of the above, plus displays files that are not candidate files, and their attributes.

For examples and more information on this flag, see the section: *The mmapplypolicy -L command in General Parallel File System: Problem Determination Guide*.

-M Name=Value...

Indicates a string substitution that will be made in the text of the policy rules before the rules are interpreted. This allows the administrator to reuse a single policy rule file for incremental backups without editing the file for each backup.

-S snapshotName

Specifies the name of a snapshot. The name appears as a subdirectory of the **.snapshots** directory in the file system root.

-s WorkDirectory

Specifies the directory to be used for temporary storage during **mmapplypolicy** command processing.

The default directory is **/tmp**. The **mmapplypolicy** command stores lists of candidate and chosen files in temporary files within this directory.

When you execute **mmapplypolicy**, it creates several temporary files and file lists. If the specified file system or directories contain many files, this can require a significant amount of temporary storage. The required storage is proportional to the number of files (NF) being acted on and the average length of the path name to each file (AVPL). To make a rough estimate of the space

required, estimate NF and assume an AVPL of 80 bytes. With an AVPL of 80, the space required is roughly (300 X NF) bytes of temporary space. Although **/tmp** is the default path name to a temporary directory, an alternative path name can be specified using the **-s** command line option.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmapplypolicy** command.

You may issue the **mmapplypolicy** command from any node in the GPFS cluster that has the file systems mounted.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. This command displays the actions that would occur if a policy were applied, but does not apply the policy at this time:

```
mmapplypolicy fs1 -P policyfile -I test
```

The system displays output similar to:

```
GPFS Current Data Pool Utilization in KB and %
sp1      2560    9765632 0.026214%
system  154880  9765632 1.585970%
Loaded policy rules from policyfile.
Evaluating MIGRATE/DELETE/EXCLUDE rules with CURRENT_TIMESTAMP = 2007-06-27@14:38:40 UTC
parsed 0 Placement Rules, 0 Restore Rules, 3 Migrate/Delete/Exclude Rules
/* Exclusion rule */
RULE 'exclude *.save files' EXCLUDE WHERE NAME LIKE '%.save'

/* Deletion rule */
RULE 'delete' DELETE FROM POOL 'sp1' WHERE NAME LIKE '%tmp%'

/* Migration rule */
RULE 'migration to system pool' MIGRATE FROM POOL 'sp1' TO POOL 'system' WHERE NAME LIKE '%file%'

Directories scan: 7 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
Inodes scan: 7 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
Summary of Rule Applicability and File Choices:
Rule#  Hit_Cnt Chosen  KB_Chosen  KB_Ill  Rule
0      2      0      0      0      RULE 'exclude *.save files' EXCLUDE WHERE(.)
1      2      2      16      0      RULE 'delete' DELETE FROM POOL 'sp1' WHERE(.)
2      2      2      16      0      RULE 'migration to system pool' MIGRATE FROM POOL
                                'sp1' TO POOL 'system' WHERE(.)
```

Files with no applicable rules: 2.

GPFS Policy Decisions and File Choice Totals:

Chose to migrate 16KB: 2 of 2 candidates;

Chose to premigrate 0KB: 0 candidates;

```

Already co-managed 0KB: 0 candidates;
Chose to delete 16KB: 2 of 2 candidates;
0KB of chosen data is illplaced or illreplicated;
Predicted Data Pool Utilization in KB and %:
sp1      2528    9765632 0.025887%
system  154896  9765632 1.586134%

```

2. This command applies a policy immediately:

```
mmapplypolicy fsl -P policyfile
```

The system displays output similar to:

```

GPFS Current Data Pool Utilization in KB and %
sp1      2560    9765632 0.026214%
system  154624  9765632 1.583349%
Loaded policy rules from policyfile.
Evaluating MIGRATE/DELETE/EXCLUDE rules with CURRENT_TIMESTAMP = 2007-06-27@15:03:22 UTC
parsed 0 Placement Rules, 0 Restore Rules, 3 Migrate/Delete/Exclude Rules
/* Exclusion rule */
RULE 'exclude *.save files' EXCLUDE WHERE NAME LIKE '%.save'

/* Deletion rule */
RULE 'delete' DELETE FROM POOL 'sp1' WHERE NAME LIKE '%tmp%'

/* Migration rule */
RULE 'migration to system pool' MIGRATE FROM POOL 'sp1' TO POOL 'system' WHERE NAME LIKE '%file%'

Directories scan: 7 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
Inodes scan: 7 files, 1 directories, 0 other objects, 0 'skipped' files and/or errors.
Summary of Rule Applicability and File Choices:
Rule#  Hit_Cnt Chosen  KB_Chosen  KB_Ill Rule
0      2        0      0          0  RULE 'exclude *.save files' EXCLUDE WHERE(.)
1      2        2      16         0  RULE 'delete' DELETE FROM POOL 'sp1' WHERE(.)
2      2        2      16         0  RULE 'migration to system pool' MIGRATE FROM POOL
                                'sp1' TO POOL 'system' WHERE(.)

```

Files with no applicable rules: 2.

```

GPFS Policy Decisions and File Choice Totals:
Chose to migrate 16KB: 2 of 2 candidates;
Chose to premigrate 0KB: 0 candidates;
Already co-managed 0KB: 0 candidates;
Chose to delete 16KB: 2 of 2 candidates;
0KB of chosen data is illplaced or illreplicated;
Predicted Data Pool Utilization in KB and %:
sp1      2528    9765632 0.025887%
system  154640  9765632 1.583512%
A total of 4 files have been migrated and/or deleted; 0 'skipped' files and/or errors.

```

3. This example shows how the **-M Name=Value** string substitution functions.

In this example the string that will be assigned to *Name* is *hsm_server* and the string that will be assigned to *Value* is *k164n04.kgn.ibm.com*. This example also uses the following policy file:

```

RULE EXTERNAL POOL 'hsm' EXEC '/var/mmfs/etc/dsmcmd' OPTS '-server hsm_server'

RULE 'move data to hsm' MIGRATE TO POOL 'hsm' WHERE name like '%tf%'

```

Using this input, when you run **mmapplypolicy -M**, GPFS scans the policy file for instances of *hsm_server* and substitutes *k164n04.kgn.ibm.com* for that string.

Therefore, running the command:

```
mmapplypolicy fsl -P policyfile -M "hsm_server=k164n04.kgn.ibm.com" -I test
```

Produces output similar to:

```

GPFS Current Data Pool Utilization in KB and %
system  165376  17795840 0.929296%
Loaded policy rules from policyfile.

```

Evaluating MIGRATE/DELETE/EXCLUDE rules with CURRENT_TIMESTAMP = 2007-06-20@18:44:26 UTC
parsed 0 Placement Rules, 0 Restore Rules, 1 Migrate/Delete/Exclude Rules

```
RULE EXTERNAL POOL 'hsm' EXEC '/var/mmfs/etc/dsmcmd' OPTS '-server k164n04.kgn.ibm.com'
```

```
RULE 'move data to hsm' MIGRATE TO POOL 'hsm' WHERE name like '%tf%'
```

```
.  
.  
.
```

4. Additional examples of GPFS policies and using the **mmapplypolicy** command are in the chapter *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

See also

“mmchpolicy Command” on page 122

“mmcrsnapshot Command” on page 142

“mmlspolicy Command” on page 230

“mmlssnapshot Command” on page 235

“mmsnapdir Command” on page 277

Location

/usr/lpp/mmfs/bin

mmauth Command

Manages secure access to GPFS file systems.

Synopsis

mmauth genkey {new | commit}

Or,

mmauth add *RemoteClusterName* **-k** *KeyFile* **-I** *CipherList*

Or,

mmauth update *RemoteClusterName* **-C** *NewClusterName* **-k** *KeyFile* [**-I** *CipherList*]

Or,

mmauth delete {*RemoteClusterName* | all }

Or,

mmauth grant {*RemoteClusterName* | all } **-f** { *Device* | all } [**-a** {rw | ro}] [**-r** {*uid:gid* | no}]

Or,

mmauth deny {*RemoteClusterName* | all } **-f** { *Device* | all }

Or,

mmauth show [*RemoteClusterName* | all]

Description

The **mmauth** command prepares a cluster to grant secure access to file systems owned locally. The **mmauth** command also prepares a cluster to receive secure access to file systems owned by another cluster. Use the **mmauth** command to generate a public/private key pair for the local cluster. A public/private key pair must be generated on both the cluster owning the file system and the cluster desiring access to the file system. The administrators of the clusters are responsible for exchanging the public portion of the public/private key pair. Use the **mmauth** command to add or delete permission for a cluster to mount file systems owned by the local cluster.

- | Remote file system access is not supported for clusters that contain Windows nodes.

When a cluster generates a new public/private key pair, administrators of clusters participating in remote file system mounts are responsible for exchanging their respective public key file **/var/mmfs/ssl/id_rsa.pub** generated by this command.

The administrator of a cluster desiring to mount a file system from another cluster must provide the received key file as input to the **mmremotecenter** command. The administrator of a cluster allowing another cluster to mount a file system must provide the received key file to the **mmauth** command.

The keyword appearing after **mmauth** determines which action is performed:

add Adds a cluster and its associated public key to the list of clusters authorized to connect to this cluster for the purpose of mounting file systems owned by this cluster.

delete Deletes a cluster and its associated public key from the list of clusters authorized to mount file systems owned by this cluster.

deny Denies a cluster the authority to mount a specific file system owned by this cluster.

genkey {new | commit}

new Generates a new public/private key pair for this cluster. The key pair is placed in */var/mmfs/ssl*. This must be done at least once before **cipherList**, the GPFS configuration parameter that enables GPFS with OpenSSL, is set.

The new key is in addition to the currently in effect committed key. Both keys are accepted until the administrator runs **mmauth genkey commit**.

commit

Commits the new public/private key pair for this cluster. Once **mmauth genkey commit** is run, the old key pair will no longer be accepted, and remote clusters that have not updated their keys (by running **mmauth update** or **mmremoteccluster update**) will be disconnected.

grant Allows a cluster to mount a specific file system owned by this cluster.

show Shows the list of clusters authorized to mount file system owned by this cluster.

update

Updates the public key and other information associated with a cluster authorized to mount file systems owned by this cluster.

When the local cluster name (or '.') is specified, **mmauth update -l** can be used to set the *cipherList* value for the local cluster. Note that you cannot use this command to change the name of the local cluster. Use the **mmchcluster** command for this purpose.

Parameters

RemoteClusterName

Specifies the remote cluster name requesting access to local GPFS file systems. The value **all** indicates all remote clusters defined to the local cluster.

Options

-a {rw | ro}

The type of access allowed:

ro Specifies read-only access.

rw Specifies read/write access. This is the default.

-C NewClusterName

Specifies a new, fully-qualified cluster name for the already-defined cluster *remoteClusterName*.

-f Device

The device name for a file system owned by this cluster. The *Device* argument is required. If **all** is specified, the command applies to all file systems owned by this cluster at the time that the command is issued.

-k KeyFile

Specifies the public key file generated by the **mmauth** command in the cluster requesting to remotely mount the local GPFS file system.

-l CipherList

Specifies the cipher list to be associated with the cluster specified by *remoteClusterName*, when connecting to this cluster for the purpose of mounting file systems owned by this cluster.

When switching from a non-secure to a secure environment and vice versa, GPFS must be down on all nodes in the cluster.

See the Frequently Asked Questions at: publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for a list of the ciphers supported by GPFS.

-r {uid:gid | no}

Specifies a root credentials remapping (*root squash*) option. The UID and GID of all processes with root credentials from the remote cluster will be remapped to the specified values. The default is not to remap the root UID and GID. The *uid* and *gid* must be specified as unsigned integers or as symbolic names that can be resolved by the operating system to a valid UID and GID. Specifying **no**, **off**, or **DEFAULT** turns off the remapping.

For more information, see *General Parallel File System: Advanced Administration Guide* and search on *root squash*.

Exit status

0 Successful completion. After a successful completion of the **mmauth** command, the configuration change request will have been propagated to all nodes in the cluster.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmauth** command.

You may issue the **mmauth** command from any node in the GPFS cluster.

Examples

1. This is an example of an **mmauth genkey new** command:

```
mmauth genkey new
```

The output is similar to this:

```
Generating RSA private key, 512 bit long modulus
.....+++++.+++++
e is 65537 (0x10001)
mmauth: Command successfully completed
mmauth: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

2. This is an example of an **mmauth genkey commit** command:

```
mmauth genkey commit
```

The output is similar to this:

```
mmauth: Command successfully completed
mmauth: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

3. This is an example of an **mmauth add** command:

```
mmauth add clustA.kgn.ibm.com -k /u/admin/keys/clustA.pub
```

The output is similar to this:

```
mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.
```

4. This is an example of an **mmauth update** command:

```
mmauth update clustA.kgn.ibm.com -k /u/admin/keys/clustA_new.pub
```

The output is similar to this:

mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.

5. This is an example of an **mmauth grant** command:

```
mmauth grant clustA.kgn.ibm.com -f /dev/gpfs1 -a ro
```

The output is similar to this:

mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.

6. This is an example on how to set or change the cipher list for the local cluster:

```
mmauth update . -l NULL-SHA
```

The output is similar to this:

mmauth: Command successfully completed
mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.

7. This is an example of an **mmauth show** command:

```
mmauth show all
```

The output is similar to this:

```
Cluster name:      clustA.kgn.ibm.com
Cipher list:       NULL-SHA
SHA digest:        a3917c8282fca7a27d951566940768dcd241902b
File system access: gpfs1 (ro)

Cluster name:      clustB.kgn.ibm.com (this cluster)
Cipher list:       NULL-SHA
SHA digest:        6ba5e3c1038246fe30f3fc8c1181fbb2130d7a8a
SHA digest (new):  3c1038246fe30f3fc8c1181fbb2130d7a8a9ab4d
File system access: (all rw)
```

For **clustB.kgn.ibm.com**, the **mmauth genkey new** command has been issued, but the **mmauth genkey commit** command has not yet been issued.

For more information on the SHA digest, see *General Parallel File System: Problem Determination Guide* and search on *SHA digest*.

8. This is an example of an **mmauth deny** command:

```
mmauth deny clustA.kgn.ibm.com -f all
```

The output is similar to this:

mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.

9. This is an example of an **mmauth delete** command:

```
mmauth delete all
```

The output is similar to this:

mmauth: Propagating the changes to all affected nodes.
This is an asynchronous process.

See also

“mmremotefs Command” on page 256

“mmremoteccluster Command” on page 253

Accessing GPFS file systems from other GPFS clusters in *General Parallel File System: Advanced Administration Guide*.

Location

`/usr/lpp/mmfs/bin`

mmbackup Command

Backs up a GPFS file system to a backup server.

Synopsis

mmbackup *Device* **-n** *ControlFile* [**-t** {**full** | **incremental**}] [**-s** *SortDir*]

Or,

mmbackup *Device* **-R** [**-s** *SortDir*]

Description

Use the **mmbackup** command to backup a GPFS file system to a backup server. **mmbackup** takes a temporary snapshot named **.mmbuSnapshot** of the specified file system, and backs up this snapshot to a back end data store. Accordingly, the files backed up by the command will be stored in the directory */Device/.snapshots/.mmbuSnapshot* in the remote data store. This command may be issued from any GPFS node in the cluster to which the file system being backed up belongs, and on which the file system is mounted.

| This command cannot be run from a Windows node.

Parameters

Device

| The device name for the file system to be backed up. File system names need not be
| fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

-n *ControlFile*

Specifies the file containing the backup control information. The file must be in the present working directory or the path must be fully qualified. Each piece of control information must be on a separate line and correctly qualified. Comment lines are allowed and must begin with a **#** in column 1. Empty lines may not contain any blank characters. Valid lines either contain a **#** in column 1 indicating a comment, an **=** indicating a value is being set, or no characters at all.

This option may be specified only if the backup type is **full** or **incremental**. If the **-R** option has been specified, this information is obtained from the control information specified on the earlier full or incremental **mmbackup** command that completed with partial success.

The allowable qualifiers in the control file are:

serverName

The name of the node specified as the backup server qualified with **serverName=**. The backup server node may or may not be a GPFS node, although performance may be improved if it is also a backup client node. You may specify only one backup server.

clientName

The backup clients, one per line, qualified with **clientName=**. The backup client nodes must be a member of the GPFS cluster where the file system is mounted. For improved performance it is suggested that multiple backup client nodes be specified. The maximum number of backup clients supported is 32.

numberOfProcessesPerClient

The number of processes per client qualified with **numberOfProcessesPerClient=**. The number of processes per client may be specified only once.

Options

-R Indicates to resume the previous backup that failed with a return code of 1 (partial success). If the

previous backup failed with a return code of 2 or succeeded with a return code of 0, this option does not succeed and a new full or incremental backup must be initiated.

-s SortDir

Specifies the directory to be used by the **sort** command for temporary data. The default is **/tmp**.

-t {full | incremental}

Specifies whether to perform a full backup of all of the files in the file system, or an incremental backup of only those files that have changed since the last backup was performed. The default is to perform an incremental backup. The default is **-t incremental**.

Exit status

- 0** Successful completion.
- 1** Partially successful completion. Not all of the eligible files were successfully backed up. The command may be resumed by specifying the **-R** option.
- 2** A failure occurred that cannot be corrected by resuming the backup. A new backup must be initiated.

Security

You must have root authority to run the **mmbackup** command.

You may issue the **mmbackup** command from any node in the cluster where the file system being backed up is mounted.

When using the **rcp** and **rsh** commands for remote communication, a properly-configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

These examples use a control file named **inputctrl32**, which specifies a backup server, three backup clients, and two processes per client as shown here:

```
# backup server
serverName=k145n06.kgn.ibm.com
# backup clients
clientName=k14n04.kgn.ibm.com
clientName=k14n05.kgn.ibm.com
clientName=k14n06.kgn.ibm.com
# number of processes per client
numberOfProcessesPerClient=2
```

1. To perform a full backup of the file system **/dev/fs0** from node **k145n04**, issue this command:

```
mmbackup /dev/fs0 -n inputctrl32 -t full
```

The system displays information similar to:

```
Tue Mar 18 14:03:25 est 2003 mmbackup is still running ...
Tue Mar 18 14:05:55 est 2003 mmbackup is still running ...
tsbackup: full backup finished with complete successs, rc = 0
```

```
mmbackup: Command successfully completed
```

2. To perform an incremental backup of the file system **/dev/fs0** from node **k145n04**, issue this command:

```
mmbbackup /dev/fs0 -n inputctrl32 -t incremental
```

The system displays information similar to:

```
Tue Mar 18 14:16:15 est 2003 mmbbackup is still running ...
tsbackup: incremental backup finished with complete success,
        rc = 0
```

```
mmbbackup: Command successfully completed
```

3. In an unsuccessful attempt to perform a full backup of the file system **/dev/fs0** from node **k145n04**, where the user had issued this command:

```
mmbbackup /dev/fs0 -n inputctrl32 -t full
```

the system displays information similar to:

```
k145n04.kgn.ibm.com: Cannot open /fs0/.mmbuTrans2.
Process 2 on client k145n06 failed in processing its list of
        files.
k145n04.kgn.ibm.com: Cannot open /fs0/.mmbuTrans5.
Process 5 on client k145n06 failed in processing its list of
        files.
tsbackup: full backup finished with partial success, rc = 1
```

```
mmbbackup: 6027-1639 Command failed. Examine previous error
        messages to determine the cause.
```

4. To resume the job unsuccessfully completed in example 3, issue this command:

```
mmbbackup /dev/fs0 -R
tsbackup: resume of full backup finished with complete success,
        rc=0
```

```
mmbbackup: Command successfully completed
```

Location

/usr/lpp/mmfs/bin

mmchattr Command

Changes the replication attributes, storage pool assignment, and I/O caching policy for one or more GPFS files.

Synopsis

mmchattr [-m *MetadataReplicas*] [-M *MaxMetadataReplicas*] [-r *DataReplicas*] [-R *MaxDataReplicas*] [-P *DataPoolName*] [-D {yes | no}] [-I {yes | defer}] *Filename* [*Filename...*]

Description

Use the **mmchattr** command to change the replication attributes, storage pool assignment, and I/O caching policy for files in the GPFS file system.

The replication factor must be less than or equal to the maximum replication factor for the file. If insufficient space is available in the file system to increase the number of replicas to the value requested, the **mmchattr** command ends. However, some blocks of the file may have their replication factor increased after the **mmchattr** command ends. If additional free space becomes available in the file system at a later time (when, for example, you add another disk to the file system), you can then issue the **mmrestripefs** command with the **-r** or **-b** option to complete the replication of the file. The **mmrestripefile** command can be used in a similar manner. You can use the **mmlsattr** command to display the replication values.

Data of a file is stored in a specific storage pool. A storage pool is a collection of disks or RAID's with similar properties. Because these storage devices have similar properties, you can manage them as a groups. You can use storage pools to:

- Partition storage for the file system
- Assign file storage locations
- Improve system performance
- Improve system reliability

The Direct I/O caching policy bypasses file cache and transfers data directly from disk into the user space buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Applications with poor cache hit rates or very large I/Os may benefit from the use of Direct I/O.

The **mmchattr** command can be run against a file in use.

You must have write permission for the files whose attributes you are changing.

Parameters

Filename [*Filename ...*]

The name of one or more files to be changed. Delimit each file name by a space. Wildcard characters are supported in file names, for example, **project*.sched**.

Options

-D {yes | no}

Enable or disable the Direct I/O caching policy for files.

-I {yes | defer}

Specifies if replication and migration between pools is to be performed immediately (**-I yes**), or deferred until a later call to **mmrestripefs** or **mmrestripefile** (**-I defer**). By deferring the updates to more than one file, the data movement may be done in parallel. The default is **-I yes**.

-m *MetadataReplicas*

Specifies how many copies of the file system's metadata to create. Enter a value of 1 or 2, but not greater than the value of the *MaxMetadataReplicas* attribute of the file.

-M *MaxMetadataReplicas*

Specifies the maximum number of copies of indirect blocks for a file. Space is reserved in the inode for all possible copies of pointers to indirect blocks. Enter a value of 1 or 2. This value cannot be less than the value of the *DefaultMetadataReplicas* attribute of the file.

-P *DataPoolName*

Changes the file's assigned storage pool to the specified *DataPoolName*. The caller must have superuser or root privileges to change the assigned storage pool.

-r *DataReplicas*

Specifies how many copies of the file data to create. Enter a value of 1 or 2. This value should not be greater than the value of the *MaxDataReplicas* attribute of the file.

-R *MaxDataReplicas*

Specifies the maximum number of copies of data blocks for a file. Space is reserved in the inode and indirect blocks for all possible copies of pointers to data blocks. Enter a value of 1 or 2. This value should not be less than the value of the *DefaultDataReplicas* attribute of the file.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have write access to the file to run the **mmchattr** command.

You may issue the **mmchattr** command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To change the metadata replication factor to 2 and the data replication factor to 2 for the **project7.resource** file in file system **fs1**, issue this command:

```
mmchattr -m 2 -r 2 /fs1/project7.resource
```

To confirm the change, issue this command:

```
mmfslattr project7.resource
```

The system displays information similar to:

```
replication factors
metadata(max) data(max) file    [flags]
-----
2 ( 2) 2 ( 2) /fs1/project7.resource
```

2. Migrating data from one storage pool to another using the **mmchattr** command with the **-I defer** option, or the **mmapplypolicy** command with the **-I defer** option will cause the data to be ill-placed. This means that the storage pool assignment for the file has changed, but the file data has not yet been migrated to the assigned storage pool.

The **mmchattr -L** command will show ill-placed flags on the files that are ill-placed. The **mmrestripefs**, or **mmrestripefile** command can be used to migrate data to the correct storage pool, and the ill-placed flag will be cleared. This is an example of an ill-placed file:

```
mmfslattr -L file1
```

The system displays output similar to this:

```
file name:          file1
metadata replication: 1 max 2
data replication:    1 max 2
flags:              illplaced
storage pool name:   spl
fileset name:        root
snapshot name:
```

See also

“mmcrfs Command” on page 130

“mmlsattr Command” on page 207

“mmlsfs Command” on page 219

Location

/usr/lpp/mmfs/bin

mmchcluster Command

Changes GPFS cluster configuration data.

Synopsis

mmchcluster *{[-p PrimaryServer] [-s SecondaryServer]}*

Or,

mmchcluster -p LATEST

Or,

mmchcluster *{[-r RemoteShellCommand] [-R RemoteFileCopyCommand]}*

Or,

mmchcluster -C ClusterName

Description

The **mmchcluster** command serves several purposes. You can use it to:

1. Change the primary or secondary GPFS cluster configuration server.
2. Synchronize the primary GPFS cluster configuration server.
3. Change the remote shell and remote file copy programs to be used by the nodes in the cluster.
4. Change the cluster name.
5. Specify node interfaces to be used by the GPFS administration commands.
6. The **mmchnode** command replaces the **mmchcluster -N** command for changes to node data related to the cluster configuration.

To display current system information for the cluster, issue the **mmlscluster** command.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

When issuing the **mmchcluster** command with the **-p** or **-s** options, the specified nodes must be available in order for the command to succeed. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command.

Attention: The **mmchcluster** command, when issued with either the **-p** or **-s** option, is designed to operate in an environment where the current primary and secondary cluster configuration servers are *not* available. As a result, the command can run without obtaining its regular serialization locks. To assure smooth transition to a new cluster configuration server, no other GPFS commands (**mm** commands) should be running when the command is issued, nor should any other command be issued until the **mmchcluster** command has successfully completed.

Parameters

-C ClusterName

Specifies a new name for the cluster. If the user-provided name contains dots, it is assumed to be a fully-qualified domain name. Otherwise, to make the cluster name unique, the domain of the primary configuration server will be appended to the user-provided name.

Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that if you change the name of the cluster, you should notify the administrators of all other GPFS clusters that can mount your file systems so that they can update their own environments. See the **mmauth**, **mmremoteclass**, and **mmremotefs** commands.

-p *PrimaryServer*

Change the primary server node for the GPFS cluster data. This may be specified as a short or long node name, an IP address, or a node number.

LATEST – Synchronize all of the nodes in the GPFS cluster ensuring they are using the most recently specified primary GPFS cluster configuration server. If an invocation of the **mmchcluster** command fails, you are prompted to reissue the command and specify **LATEST** on the **-p** option to synchronize all of the nodes in the GPFS cluster. Synchronization provides for all nodes in the GPFS cluster to use the most recently specified primary GPFS cluster configuration server.

-s *SecondaryServer*

Change the secondary server node for the GPFS cluster data. To remove the secondary GPFS server and continue operating without it, specify a null string, "", as the parameter. This may be specified as a short or long nodename, an IP address, or a node number.

Options

-R *RemoteFileCopy*

Specifies the fully-qualified path name for the remote file copy program to be used by GPFS.

The remote copy command must adhere to the same syntax format as the **rcp** command, but may implement an alternate authentication mechanism.

-r *RemoteShellCommand*

Specifies the fully-qualified path name for the remote shell program to be used by GPFS.

The remote shell command must adhere to the same syntax format as the **rsh** command, but may implement an alternate authentication mechanism.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchcluster** command.

You may issue the **mmchcluster** command from any node in the GPFS cluster.

A properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To change the primary GPFS server for the cluster, issue this command:

```
mmchcluster -p k164n06
```

The system displays output similar to:

```
mmchcluster: Command successfully completed
```

To confirm the change, issue this command:

```
mmiscluster
```

The system displays information similar to:

```
GPFS cluster information
```

```
=====
```

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

```
GPFS cluster configuration servers:
```

```
-----
```

```
Primary server:   k164n06.kgn.ibm.com
Secondary server: k164n05.kgn.ibm.com
```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.71	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164sn06.kgn.ibm.com	

See also

“mmaddnode Command” on page 70

“mmchnode Command” on page 116

“mmcrcluster Command” on page 124

“mmdelnode Command” on page 170

“mmiscluster Command” on page 209

“mmremoteccluster Command” on page 253

Location

/usr/lpp/mmfs/bin

mmchconfig Command

Changes GPFS configuration parameters.

Synopsis

mmchconfig *Attribute=value[,Attribute=value...]* [-i | -I] [-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}]

Description

Use the **mmchconfig** command to change the GPFS configuration attributes on a single node, a set of nodes, or globally for the entire cluster.

The *Attribute=value* options must come before any operand.

When changing both **maxblocksize** and **pagepool**, the command fails unless these conventions are followed:

- When increasing the values, **pagepool** must be specified first.
- When decreasing the values, **maxblocksize** must be specified first.

Results

The configuration is updated on each node in the GPFS cluster.

Parameters

-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}

Specifies the set of nodes to which the configuration changes apply. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

The **-N** flag is valid only for the **automountDir**, **dataStructureDump**, **designation**, **dmapiEventTimeout**, **dmapiMountTimeout**, **dmapiSessionFailureTimeout**, **maxblocksize**, **maxFilesToCache**, **maxStatCache**, **nsdServerWaitTimeWindowOnMount**, **nsdServerWaitTimeForMount**, **pagepool**, **prefetchThreads**, **unmountOnDiskFail**, and **worker1Threads** attributes.

This command does not support a *NodeClass* of **mount**.

Options

Attribute=value

Specifies the name of the attribute to be changed and its associated *value*. More than one attribute and value pair, in a comma-separated list, can be changed with one invocation of the command.

To restore the GPFS default setting for any given attribute, specify **DEFAULT** as its *value*.

- I** Specifies that the changes take effect immediately, but do not persist when GPFS is restarted. This option is valid only for the **dataStructureDump**, **dmapiEventTimeout**, **dmapiMountTimeout**, **dmapiSessionFailureTimeout**, **maxMBpS**, **pagepool**, **unmountOnDiskFail**, and **verbsRdma** attributes.
- i** Specifies that the changes take effect immediately and are permanent. This option is valid only for the **dataStructureDump**, **dmapiEventTimeout**, **dmapiMountTimeout**, **dmapiSessionFailureTimeout**, **maxMBpS**, **pagepool**, **unmountOnDiskFail**, and **verbsRdma** attributes.

autoload

Starts GPFS automatically whenever the nodes are rebooted. Valid values are **yes** or **no**.

automountDir

Specifies the directory to be used by the Linux automounter for GPFS file systems that are being mounted automatically. The default directory is **/gpfs/automountdir**. This parameter does not apply to AIX and Windows environments.

cipherList

Controls whether GPFS network communications are secured. If **cipherList** is not specified, or if the value **DEFAULT** is specified, GPFS does not authenticate or check authorization for network connections. If the value **AUTHONLY** is specified, GPFS does authenticate and check authorization for network connections, but data sent over the connection is not protected. Before setting **cipherList** for the first time, you must establish a public/private key pair for the cluster by using the **mmauth genkey new** command.

GPFS must be down on all the nodes if you are switching from a non-secure environment to a secure environment and vice versa.

Cipher lists are not supported in clusters that contain Windows nodes.

See the Frequently Asked Questions at: publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for a list of the ciphers supported by GPFS.

cnfsMountdPort

Specifies the port number to be used for **rpc.mountd**. See *General Parallel File System: Advanced Administration Guide* for restrictions and additional information.

cnfsNFSDprocs

Specifies the number of **nfds** kernel threads. The default is 32.

cnfsSharedRoot

Specifies a directory in a GPFS file system to be used by the clustered NFS subsystem.

GPFS must be down on all the nodes in the cluster when changing the **cnfsSharedRoot** attribute.

See the *General Parallel File System: Advanced Administration Guide* for restrictions and additional information.

cnfsVIP

Specifies a virtual DNS name for the list of CNFS IP addresses assigned to the nodes with the **mmchnode** command. This allows NFS clients to be distributed among the CNFS nodes using DNS round robin. For additional Information, see *General Parallel File System: Advanced Administration Guide*.

dataStructureDump

Specifies a path for the storage of dumps. The default is to store dumps in **/tmp/mmfs**. Specify **no** to not store dumps.

It is suggested that you create a directory for the placement of certain problem determination information. This can be a symbolic link to another location if more space can be found there. Do not place it in a GPFS file system, because it might not be available if GPFS fails. If a problem occurs, GPFS may write 200 MB or more of problem determination data into the directory. These files must be manually removed when problem determination is complete. This should be done promptly so that a **NOSPACE** condition is not encountered if another failure occurs.

defaultMountDir

Specifies the default parent directory for GPFS file systems. The default value is **/gpfs**. If an explicit mount directory is not provided with the **mmcrfs**, **mmchfs**, or **mmremotefs** command, the default mount point will be set to *DefaultMountDir/DeviceName*.

designation

This option is no longer valid. Use the “mmchnode Command” on page 116 instead.

dmapiEventTimeout

Controls the blocking of file operation threads of NFS, while in the kernel waiting for the handling of a DMAPI synchronous event. The parameter value is the maximum time, in milliseconds, the thread will block. When this time expires, the file operation returns **ENOTREADY**, and the event continues asynchronously. The NFS server is expected to repeatedly retry the operation, which eventually will find the response of the original event and continue. This mechanism applies only to read, write, and truncate event types, and only when such events come from NFS server threads. The timeout value is given in milliseconds. The value 0 indicates immediate timeout (fully asynchronous event). A value greater than or equal to 86400000 (which is 24 hours) is considered *infinity* (no timeout, fully synchronous event). The default value is 86400000.

For further information regarding DMAPI for GPFS, see the *General Parallel File System: Data Management API Guide*.

dmapiMountEvent

Controls the generation of the **mount**, **preunmount**, and **unmount** events. Valid values are:

all **mount**, **preunmount**, and **unmount** events are generated on each node. This is the default behavior.

SessionNode

mount, **preunmount**, and **unmount** events are generated on each node and are delivered to the session node, but the session node will not deliver the event to the DMAPI application unless the event is originated from the **SessionNode** itself.

LocalNode

mount, **preunmount**, and **unmount** events are generated only if the node is a session node.

For further information regarding DMAPI for GPFS, see the *General Parallel File System: Data Management API Guide*.

dmapiMountTimeout

Controls the blocking of **mount** operations, waiting for a disposition for the mount event to be set. This timeout is activated, at most once on each node, by the first external mount of a file system that has DMAPI enabled, and only if there has never before been a mount disposition. Any **mount** operation on this node that starts while the timeout period is active will wait for the mount disposition. The parameter value is the maximum time, in seconds, that the **mount** operation will wait for a disposition. When this time expires and there is still no disposition for the mount event, the **mount** operation fails, returning the **EIO** error. The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the mount operation). A value greater than or equal to 86400 (which is 24 hours) is considered *infinity* (no timeout, indefinite blocking until there is a disposition). The default value is 60.

For further information regarding DMAPI for GPFS, see the *General Parallel File System: Data Management API Guide*.

dmapiSessionFailureTimeout

Controls the blocking of file operation threads, while in the kernel, waiting for the handling of a DMAPI synchronous event that is enqueued on a session that has experienced a failure. The parameter value is the maximum time, in seconds, the thread will wait for the recovery of the failed session. When this time expires and the session has not yet recovered, the event is cancelled and the file operation fails, returning the **EIO** error. The timeout value is given in full seconds. The value 0 indicates immediate timeout (immediate failure of the file operation). A value greater than or equal to 86400 (which is 24 hours) is considered *infinity* (no timeout, indefinite blocking until the session recovers). The default value is 0.

For further information regarding DMAPI for GPFS, see the *General Parallel File System: Data Management API Guide*.

failureDetectionTime

Used when persistent reserve (PR) is enabled and it indicates to GPFS the amount of time it will take to detect that a node has failed.

GPFS must be down on all the nodes when changing the **failureDetectionTime** attribute.

maxblocksize

Changes the maximum file system block size. Valid values are 64 KB, 256 KB, 512 KB, 1 MB, 2 MB, and 4 MB. The default value is 1 MB. Specify this value with the character **K** or **M**, for example 512K.

File systems with block sizes larger than the specified value cannot be created or mounted unless the block size is increased.

GPFS must be down on all the nodes when changing the **maxblocksize** attribute.

maxFilesToCache

Specifies the number of inodes to cache for recently used files that have been closed.

Storing a file's inode in cache permits faster re-access to the file. The default is 1000, but increasing this number may improve throughput for workloads with high file reuse. However, increasing this number excessively may cause paging at the file system manager node. The value should be large enough to handle the number of concurrently open files plus allow caching of recently used files.

maxMBps

Specifies an estimate of how many megabytes of data can be transferred per second into or out of a single node. The default is 150 MB per second. The value is used in calculating the amount of I/O that can be done to effectively prefetch data for readers and write-behind data from writers. By lowering this value, you can artificially limit how much I/O one node can put on all of the disk servers.

This is useful in environments in which a large number of nodes can overrun a few virtual shared disk servers. Setting this value too high usually does not cause problems because of other limiting factors, such as the size of the pagepool, the number of prefetch threads, and so forth.

maxStatCache

Specifies the number of inodes to keep in the stat cache. The stat cache maintains only enough inode information to perform a query on the file system. The default value is:

4 × maxFilesToCache

nsdServerWaitTimeForMount

When mounting a file system whose disks depend on NSD servers, this option specifies the number of seconds to wait for those servers to come up. The decision to wait is controlled by the criteria managed by the **nsdServerWaitTimeWindowOnMount** option.

Valid values are between 0 and 1200 seconds. The default is 300. A value of zero indicates that no waiting is done. The interval for checking is 10 seconds. If **nsdServerWaitTimeForMount** is 0, **nsdServerWaitTimeWindowOnMount** has no effect.

The mount thread waits when the daemon delays for safe recovery. The mount wait for NSD servers to come up, which is covered by this option, occurs after expiration of the recovery wait allows the mount thread to proceed.

nsdServerWaitTimeWindowOnMount

Specifies a window of time (in seconds) during which a mount can wait for NSD servers as described for the **nsdServerWaitTimeForMount** option. The window begins when quorum is established (at cluster startup or subsequently), or at the last known failure times of the NSD servers required to perform the mount.

Valid values are between 1 and 1200 seconds. The default is 600. If **nsdServerWaitTimeForMount** is 0, **nsdServerWaitTimeWindowOnMount** has no effect.

When a node rejoins the cluster after having been removed for any reason, the node resets all the failure time values that it knows about. Therefore, when a node rejoins the cluster it believes that the NSD servers have not failed. From the node's perspective, old failures are no longer relevant.

GPFS checks the cluster formation criteria first. If that check falls outside the window, GPFS then checks for NSD server fail times being within the window.

pagepool

Changes the size of the cache on each node. The default value is 64 MB. The minimum allowed value is 4 MB. The maximum GPFS pagepool size can be as large as 256 GB on 64-bit Linux systems and 64-bit AIX systems. Specify this value with the character **M**, for example, 128M.

pagepoolMaxPhysMemPct

Percentage of physical memory that can be assigned to the page pool. Valid values are between 10 and 90 percent. The default is 50 percent.

prefetchThreads

Controls the maximum possible number of threads dedicated to prefetching data for files that are read sequentially, or to handle sequential write-behind.

Functions in the GPFS daemon dynamically determine the actual degree of parallelism for prefetching data. The default value is 72. The minimum value is 2. The maximum value of **prefetchThreads** plus **worker1Threads** is:

- 164 on 32-bit kernels
- 550 on 64-bit kernels

release=LATEST

Changes the GPFS configuration information to the latest format supported by the currently installed level of GPFS. Perform this operation after all nodes in the GPFS cluster have been migrated to the latest level of GPFS. For additional information see *Completing the migration to a new level of GPFS* in the *General Parallel File System: Concepts, Planning and Installation Guide*.

This command attempts to access each of the nodes in the cluster to verify the level of the installed GPFS code. If one or more nodes cannot be reached, you will have to rerun the command until the information for all nodes can be confirmed.

sidAutoMapRangeLength

Controls the length of the reserved range for Windows SID to UNIX ID mapping. See *Identity management on Windows* in the *General Parallel File System: Advanced Administration Guide* for additional information.

sidAutoMapRangeStart

Specifies the start of the reserved range for Windows SID to UNIX ID mapping. See *Identity management on Windows* in the *General Parallel File System: Advanced Administration Guide* for additional information.

subnets

Specifies subnets used to communicate between nodes in a GPFS cluster or a remote GPFS cluster.

The subnets option must use the following format:

```
subnets="Subnet[/ClusterName[;ClusterName...][ Subnet[/ClusterName[;ClusterName...]]...]"
```

The order in which you specify the subnets determines the order that GPFS uses these subnets to establish connections to the nodes within the cluster. For example, **subnets="192.168.2.0"** refers to IP addresses 192.168.2.0 through 192.168.2.255.

This feature cannot be used to establish fault tolerance or automatic failover. If the interface corresponding to an IP address in the list is down, GPFS does not use the next one on the list. For more information about subnets, see *General Parallel File System: Advanced Administration Guide* and search on *Using remote access with public and private IP addresses*.

tiebreakerDisks

Controls whether GPFS will use the node quorum with tiebreaker algorithm in place of the regular node based quorum algorithm. See *General Parallel File System: Concepts, Planning, and Installation Guide* and search for *node quorum with tiebreaker*. To enable this feature, specify the names of one or three disks. Separate the NSD names with semicolon (;) and enclose the list in quotes. The disks do not have to belong to any particular file system, but must be directly accessible from the quorum nodes. For example:

```
tiebreakerDisks="gpfs1nsd;gpfs2nsd;gpfs3nsd"
```

To disable this feature, use:

```
tiebreakerDisks=no
```

When changing the **tiebreakerDisks**, GPFS must be down on all nodes in the cluster.

uidDomain

Specifies the UID domain name for the cluster.

GPFS must be down on all the nodes when changing the **uidDomain** attribute.

A detailed description of the GPFS user ID remapping convention is contained in the *UID Mapping for GPFS in a Multi-Cluster Environment* white paper at http://www.ibm.com/systems/clusters/library/wp_lit.html.

unmountOnDiskFail

Controls how the GPFS daemon will respond when a disk failure is detected. Valid values are **yes** or **no**.

When **unmountOnDiskFail** is set to **no**, the daemon marks the disk as failed and continues as long as it can without using the disk. All nodes that are using this disk are notified of the disk failure. The disk can be made active again by using the **mmchdisk** command. This is the suggested setting when metadata and data replication are used because the replica can be used until the disk is brought online again.

When **unmountOnDiskFail** is set to **yes**, any disk failure will cause only the local node to force-unmount the file system that contains that disk. Other file systems on this node and other nodes continue to function normally, if they can. The local node can try and remount the file system when the disk problem has been resolved. This is the suggested setting when using SAN-attached disks in large multinode configurations, and when replication is not being used. This setting should also be used on a node that hosts **descOnly** disks. See *Establishing disaster recovery for your GPFS cluster* in *General Parallel File System: Advanced Administration Guide*.

usePersistentReserve

Specifies whether to enable or disable Persistent Reserve (PR) on the disks. Valid values are **yes** or **no** (**no** is the default). GPFS must be stopped on all nodes when setting this attribute.

- PR is only supported on AIX nodes.
- PR is only supported on NSDs that are built directly on hdisks.
- The disk subsystem must support PR
- GPFS supports a mix of PR disks and other disks. However, you will only realize improved failover times if **all** the disks in the cluster support PR.
- GPFS only supports PR in the home cluster. Remote mounts must access the disks using an NSD server.

For more information, see *Reduced recovery time using Persistent Reserve* in the *General Parallel File System: Concepts, Planning, and Installation Guide*.

verbsPorts

Specifies the InfiniBand device names and port numbers used for RDMA transfers between an NSD client and server. You must enable **verbsRdma** to enable **verbsPorts**.

The format for verbsPorts is:

```
verbsPorts="device/port[ device/port ...]"
```

In this format, *device* is the HCA device name (such as *mtxca0*) and *port* is the one-based port number (such as 1 or 2). If you do not specify a port number, GPFS uses port 1 as the default.

For example:

```
verbsPorts="mtxca0/1 mtxca0/2"
```

will create two RDMA connections between the NSD client and server using both ports of a dual ported adapter.

verbsRdma

Enables or disables InfiniBand RDMA using the Verbs API for data transfers between an NSD client and NSD server. Valid values are **enable** or **disable**.

Note: InfiniBand RDMA for Linux X86_64 is supported only on GPFS V3.2 Multiplatform. For the latest support information, see the GPFS Frequently Asked Questions at:
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

worker1Threads

Controls the maximum number of concurrent file operations at any one instant. If there are more requests than that, the excess will wait until a previous request has finished.

This attribute is primarily used for random read or write requests that cannot be pre-fetched, random I/O requests, or small file activity. The default value is 48. The minimum value is 1. The maximum value of **prefetchThreads** plus **worker1Threads** is:

- 164 on 32-bit kernels
- 550 on 64-bit kernels

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchconfig** command.

You may issue the **mmchconfig** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To change the maximum file system block size allowed to 4 MB, issue this command:

```
mmchconfig maxblocksize=4M
```

To confirm the change, issue this command:

```
mmfsconfig
```

The system displays information similar to:

Configuration data for cluster cluster1.kgn.ibm.com:

```
-----  
clusterName cluster1.kgn.ibm.com  
clusterId 680681562214606028  
clusterType lc  
autoload yes  
minReleaseLevel 3.2.0.0  
pagepool 512m  
maxblocksize 4M  
cipherList AUTHONLY
```

File systems in cluster cluster1.kgn.ibm.com:

```
-----  
/dev/fs1  
/dev/fs2
```

See also

“mmaddnode Command” on page 70

“mmchnode Command” on page 116

“mmcrcluster Command” on page 124

“mmdelnode Command” on page 170

“mmlsconfig Command” on page 211

“mmlscluster Command” on page 209

Location

/usr/lpp/mmfs/bin

mmchdisk Command

Changes state or parameters of one or more disks in a GPFS file system.

Synopsis

```
mmchdisk Device {suspend | resume | stop | start | change} -d "DiskDesc [;DiskDesc...]" | -F {DescFile} [-N {Node [,Node...] | NodeFile | NodeClass}]
```

Or,

```
mmchdisk Device {resume | start} -a [[-N {Node [,Node...] | NodeFile | NodeClass}]
```

Description

Use the **mmchdisk** command to change the state or the parameters of one or more disks in a GPFS file system.

The state of a disk is a combination of its status and availability, displayed with the **mmldisk** command. Disk status is normally either **ready** or **suspended**. A transitional status such as **replacing**, **replacement**, or **being emptied** might also appear if a disk is being deleted or replaced. A suspended disk is one that the user has decided not to place any new data on. Existing data on a suspended disk may still be read or updated. Typically, a disk is suspended prior to restriping a file system. Suspending a disk tells the **mmrestripefs** command that data is to be migrated off that disk. Disk availability is either **up** or **down**.

Be sure to use **stop** before you take a disk offline for maintenance. You should also use **stop** when a disk has become temporarily inaccessible due to a disk failure that is repairable without loss of data on that disk (for example, an adapter failure or a failure of the disk electronics).

The *Disk Usage* (**dataAndMetadata**, **dataOnly**, **metadataOnly**, or **descOnly**) and *Failure Group* parameters of a disk are adjusted with the **change** option. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search for *recoverability considerations*. The **mmchdisk change** command does not move data or metadata that resides on the disk. After changing disk parameters, in particular, *Disk Usage*, you may have to issue the **mmrestripefs** command with the **-r** option to relocate data so that it conforms to the new disk parameters.

The **mmchdisk** command can be issued for a mounted or unmounted file system. When maintenance is complete or the failure has been repaired, use the **mmchdisk** command with the **start** option. If the failure cannot be repaired without loss of data, you can use the **mmdeldisk** command.

Note:

1. The **mmchdisk** command cannot be used to change the NSD servers associated with the disk. Use the **mmchnsd** command for this purpose.
2. Similarly, the **mmchdisk** command cannot be used to change the storage pool for the disk. Use the **mmdeldisk** and **mmadddisk** commands to move a disk from one storage pool to another.

Parameters

Device

The device name of the file system to which the disks belong. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

-d "*DiskDesc* [*;DiskDesc...*]"

A descriptor for each disk to be changed.

Specify only disk names when using the **suspend**, **resume**, **stop**, or **start** options. Delimit multiple disk names with semicolons and enclose the list in quotation marks. For example, "gpfs1nsd;gpfs2nsd"

When using the **change** option, include the disk name and any new *Disk Usage* and *Failure Group* positional parameter values in the descriptor. Delimit descriptors with semicolons and enclose the list in quotation marks, for example, "gpfs1nsd::dataOnly;gpfs2nsd::metadataOnly:12"

A disk descriptor is defined as (second, third, sixth and seventh fields reserved):

DiskName::DiskUsage:FailureGroup:::

DiskName

For a list of disks that belong to a particular file system, issue the **mmlsnsd -f**, the **mmlsfs -d**, or the **mmlsdisk** command. The **mmlsdisk** command will also show the current disk usage and failure group values for each of the disks.

DiskUsage

If a value is not specified, the disk usage remains unchanged:

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default.

dataOnly

Indicates that the disk contains data and does not contain metadata.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

FailureGroup

A number identifying the failure group to which this disk belongs. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. If you do not specify a failure group, the value remains unchanged. GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single disk failure. All disks that are attached to the same NSD server or adapter should be placed in the same failure group.

-F {DescFile}

Specifies a file containing a list of disk descriptors, one per line.

-a

Specifies to change the state of all of the disks belonging to the file system, *Device*. This operand is valid only on the **resume** and **start** options.

-N {Node[,Node...] | NodeFile | NodeClass }

Specify the nodes to participate in the restripe of the file system after the state or parameters of the disks have been changed. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the restripe of the file system).

For general information on how to specify node names, see "Specifying nodes as input to GPFS commands" on page 2.

Options

change

Instructs GPFS to change the *DiskUsage* parameter, the *FailureGroup* parameter, or both, according to the values specified in the *DiskDesc*.

resume

Informs GPFS that a disk previously suspended is now available for allocating new space. If the disk is currently in a stopped state, it remains stopped until you specify the **start** option. Otherwise, normal read and write access to the disk resumes.

start Informs GPFS that disks previously stopped are now accessible. This is accomplished by first changing the disk availability from **down** to **recovering**. The file system metadata is then scanned and any missing updates (replicated data that was changed while the disk was **down**) are repaired. If this operation is successful, the availability is then changed to **up**. If the metadata scan fails, availability is set to **unrecovered**. This could occur if too many other disks are **down**. The metadata scan can be re-initiated at a later time by issuing the **mmchdisk start** command again.

If more than one disk in the file system is down, they must all be started at the same time by issuing the **mmchdisk Device start -a** command. If you start them separately and metadata is stored on any disk that remains down, the **mmchdisk start** command fails.

stop Instructs GPFS to stop any attempts to access the specified disks. Use this option to tell the file system manager that a disk has failed or is currently inaccessible because of maintenance.

A disk remains stopped until it is explicitly started by the **mmchdisk** command with the **start** option. Restarting the GPFS Server daemon or rebooting does not restore normal access to a stopped disk.

suspend

Instructs GPFS to stop allocating space on the specified disk. Place a disk in this state when you are preparing to restripe the file system off this disk because of faulty performance. This is a user-initiated state that GPFS never uses without an explicit command to change disk state. Existing data on a suspended disk may still be read or updated.

A disk remains suspended until it is explicitly resumed. Restarting GPFS or rebooting nodes does not restore normal access to a suspended disk.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchdisk** command.

You may issue the **mmchdisk** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To **suspend** active disk **gpfs2nsd**, issue this command:

```
mmchdisk fs0 suspend -d gpfs2nsd
```

To confirm the change, issue this command:

```
mmllsdisk fs0
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
gpfs2nsd	nsd	512	2	yes	yes	suspended	up	system
hd3vsdn01	nsd	512	2	yes	yes	ready	up	system
hd27n01	nsd	512	8	yes	yes	ready	up	system
hd28n01	nsd	512	8	yes	yes	ready	up	system
hd29n01	nsd	512	8	yes	yes	ready	up	system
hd10vsdn09	nsd	512	4003	no	yes	ready	up	sp1
hd11vsdn10	nsd	512	4003	no	yes	ready	up	sp1

2. To specify that metadata should no longer be stored on disk **gpfs1nsd**, issue this command:

```
mmchdisk fs0 change -d "gpfs1nsd:::dataOnly"
```

To confirm the change, issue this command:

```
mmllsdisk fs0
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
hd2vsdn01	nsd	512	2	yes	yes	ready	up	system
hd3vsdn01	nsd	512	2	yes	yes	ready	up	system
hd27n01	nsd	512	8	yes	yes	ready	up	system
gpfs1nsd	nsd	512	8	no	yes	ready	up	system
hd29n01	nsd	512	8	yes	yes	ready	up	system
hd10vsdn09	nsd	512	4003	no	yes	ready	up	sp1
hd11vsdn10	nsd	512	4003	no	yes	ready	up	sp1

See also

“Displaying GPFS disk states” on page 36

“mmadddisk Command” on page 66

“mmchnsd Command” on page 120

“mmdeldisk Command” on page 162

“mmllsdisk Command” on page 213

“mmllnsd Command” on page 227

“mmrpldisk Command” on page 271

Location

/usr/lpp/mmfs/bin

mmcheckquota Command

Checks file system user, group and fileset quotas.

Synopsis

mmcheckquota [-v] {*Device* [*Device...*] | -a}

Or,

mmcheckquota {[-u *UserQuotaFilename*] | [-g *GroupQuotaFileName*] | [-j *FilesetQuotaFilename*]} *Device*

Description

The **mmcheckquota** command serves two purposes:

1. Count inode and space usage in a file system by user, group and fileset, and write the collected data into quota files.
2. Replace either the user, group, or fileset quota files, for the file system designated by *Device*, thereby restoring the quota files for the file system. These files must be contained in the root directory of *Device*. If a backup copy does not exist, an empty file is created when the **mmcheckquota** command is issued.

The **mmcheckquota** command counts inode and space usage for a file system and writes the collected data into quota files. Indications leading you to the conclusion you should run the **mmcheckquota** command include:

- **MMFS_QUOTA** error log entries. This error log entry is created when the quota manager has a problem reading or writing the quota file.
- Quota information is lost due to a node failure. A node failure could leave users unable to open files or deny them disk space that their quotas should allow.
- The in-doubt value is approaching the quota limit.

The sum of the in-doubt value and the current usage may not exceed the hard limit. Consequently, the actual block space and number of files available to the user of the group may be constrained by the in-doubt value. Should the in-doubt value approach a significant percentage of the quota, use the **mmcheckquota** command to account for the lost space and files.

- User, group, or fileset quota files are corrupted.

The **mmcheckquota** command is I/O intensive and should be run when the system load is light. When issuing the **mmcheckquota** command on a mounted file system, negative in-doubt values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

| This command cannot be run from a Windows node.

Parameters

-a Checks all GPFS file systems in the cluster from which the command is issued.

Device

The device name of the file system. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

-g *GroupQuotaFileName*

Replace the current group quota file with the file indicated.

When replacing quota files with the **-g** option:

- The quota file must be in the root directory of the file system.
- The file system must be unmounted.

-j *FilesetQuotaFilename*

Replace the current fileset quota file with the file indicated.

When replacing quota files with the **-j** option:

- The quota file must be in the root directory of the file system.
- The file system must be unmounted.

-u *UserQuotaFilename*

Replace the current user quota file with the file indicated.

When replacing quota files with the **-u** option:

- The quota file must be in the root directory of the file system.
- The file system must be unmounted.

Options

-v Reports discrepancies between calculated and recorded disk quotas.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcheckquota** command.

GPFS must be running on the node from which the **mmcheckquota** command is issued.

Examples

1. To check quotas for file system **fs0**, issue this command:

```
mmcheckquota fs0
```

The system displays information only if a problem is found.

2. To check quotas for all file systems, issue this command:

```
mmcheckquota -a
```

The system displays information only if a problem is found or if quota management is not enabled for a file system:

```
fs2: no quota management installed
fs3: no quota management installed
```

3. To report discrepancies between calculated and recorded disk quotas, issue this command:

```
mmcheckquota -v fs1
```

The system displays information similar to:

```
fs1: Start quota check
19 % complete on Mon Feb 6 17:39:55 2006
34 % complete on Mon Feb 6 17:40:26 2006
55 % complete on Mon Feb 6 17:40:57 2006
64 % complete on Mon Feb 6 17:41:29 2006
85 % complete on Mon Feb 6 17:42:00 2006
99 % complete on Mon Feb 6 17:42:33 2006
fs1: quota check found the following differences:
USR 0: 98463 subblocks counted (was 3601); 17404 inodes counted (was 15999)
USR 60001: 4766 subblocks counted (was -4499); 644 inodes counted (was 639)
USR 60008: 4162 subblocks counted (was -5654); 534 inodes counted (was 532)
```

USR 60012: 7248 subblocks counted (was -1619); 2013 inodes counted (was 2217)
USR 60013: 6915 subblocks counted (was -616); 1773 inodes counted (was 2297)
USR 60014: 6553 subblocks counted (was -1124); 1885 inodes counted (was 2533)
USR 60020: 7045 subblocks counted (was -2486); 2050 inodes counted (was 1986)
GRP 0: 98529 subblocks counted (was 6406); 17437 inodes counted (was 15910)
GRP 100: 116038 subblocks counted (was -65884); 26277 inodes counted (was 30656)
FILESET 0: 214567 subblocks counted (was -60842); 43714 inodes counted (was 46661)

See also

“mmedquota Command” on page 182

“mmfsck Command” on page 187

“mmlsquota Command” on page 232

“mmquotaon Command” on page 251

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmchfileset Command

Changes the attributes of a GPFS fileset.

Synopsis

mmchfileset *Device* {*FilesetName* | **-J** *JunctionPath*} [{**-j** *NewFilesetName*] | [**-t** *NewComment*]}

Description

The **mmchfileset** command changes the name or comment for an existing GPFS fileset.

For information on GPFS filesets, see the topic *Policy-based data management for GPFS* in the *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system that contains the fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

FilesetName

Specifies the name of the fileset.

-J *JunctionPath*

Specifies the junction path name for the fileset.

A junction is a special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

-j *NewFilesetName*

Specifies the new name that is to be given to the fileset. This name must be less than 256 characters in length. This flag may be specified along with the **-t** flag.

-t *NewComment*

Specifies an optional comment that appears in the output of the **mmisfileset** command. This comment must be less than 256 characters in length. This flag may be specified along with the **-j** flag.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchfileset** command.

You may issue the **mmchfileset** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This command renames fileset **fset1** to **fset2**, and gives it the comment "the first fileset":

```
mmchfileset gpfs1 fset1 -j fset2 -t 'the first fileset'
```

The system displays a message similar to:

```
Fileset 'fset1' changed.
```

To confirm the change, issue this command:

```
mmlsfileset gpfs1 -L
```

The system displays information similar to:

Name	Id	RootInode	ParentId	Created	Comment
root	0	3	--	Thu Dec 12 15:59:18 2005	root fileset
fset2	1	261633	0	Mon Dec 13 13:55:28 2005	the first fileset

See also

"[mmdelfileset Command](#)" on page 165

"[mmcrfileset Command](#)" on page 128

"[mmlinkfileset Command](#)" on page 205

"[mmlsfileset Command](#)" on page 216

"[mmunlinkfileset Command](#)" on page 288

Location

/usr/lpp/mmfs/bin

mmchfs Command

Changes the attributes of a GPFS file system.

Synopsis

mmchfs *Device* [-A {yes | no | automount}] [-E {yes | no}] [-D {nfs4 | posix}] [-F *MaxNumInodes[:NumInodesToPreallocate]*] [-k {posix | nfs4 | all}] [-K {no | whenpossible | always}] [-m *DefaultMetadataReplicas*] [-o *MountOptions*] [-Q {yes | no}] [-r *DefaultDataReplicas*] [-S {yes | no}] [-T *Mountpoint*] [-t *DriveLetter*] [-V {full | compat}] [-z {yes | no}]

Or,

mmchfs *Device* [-W *NewDeviceName*]

Description

Use the **mmchfs** command to change the attributes of a GPFS file system.

Parameters

Device

The device name of the file system to be changed.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**. However, file system names must be unique across GPFS clusters.

This must be the first parameter.

Options

-A {yes | no | automount}

Indicates when the file system is to be mounted:

yes When the GPFS daemon starts.

no Manual mount.

automount

On non-Windows nodes, when the file system is first accessed. On Windows nodes, when the GPFS daemon starts.

Note: The file system must be unmounted prior to changing the automount settings.

-D {nfs4 | posix}

Specifies whether a 'deny-write open lock' will block writes, which is expected and required by NFS V4. File systems supporting NFS V4 must have **-D nfs4** set. The option **-D posix** allows NFS writes even in the presence of a deny-write open lock. If you intend to export the file system using NFS V4 or Samba, you must use **-D nfs4**. For NFS V3 (or if the file system is not NFS exported at all) use **-D posix**.

-E {yes | no}

Specifies whether to report exact **mtime** values. If **-E no** is specified, the **mtime** value is periodically updated. If you desire to always display exact modification times, specify the **-E yes** option.

-F *MaxNumInodes[:NumInodesToPreallocate]*

MaxNumInodes specifies the maximum number of files that can be created. Allowable values range from the current number of created inodes (determined by issuing the **mmdf** command with the **-F** option), through the maximum number of files possibly supported as constrained by the formula:

maximum number of files = (total file system space/2) / (inode size + subblock size)

If your file system has additional disks added or the number of inodes was insufficiently sized at file system creation, you can change the number of inodes and hence the maximum number of files that can be created.

For file systems that will be doing parallel file creates, if the total number of free inodes is not greater than 5% of the total number of inodes, there is the potential for slowdown in file system access. Take this into consideration when changing your file system.

NumInodesToPreallocate specifies the number of inodes that will be pre-allocated by the system right away. If this number is not specified, GPFS allocates inodes dynamically as needed.

The *MaxNumInodes* and *NumInodesToPreallocate* values can be specified with a suffix, for example 100K or 2M.

-k {posix | nfs4 | all}

Specifies the type of authorization supported by the file system:

posix Traditional GPFS ACLs only (NFS V4 ACLs are not allowed). Authorization controls are unchanged from earlier releases.

nfs4 Support for NFS V4 ACLs only. Users are not allowed to assign traditional GPFS ACLs to any file system objects (directories and individual files).

all Any supported ACL type is permitted. This includes traditional GPFS (**posix**) and NFS V4 ACLs (**nfs4**).

The administrator is allowing a mixture of ACL types. For example, **fileA** may have a **posix** ACL, while **fileB** in the same file system may have an NFS V4 ACL, implying different access characteristics for each file depending on the ACL type that is currently assigned.

Neither **nfs4** nor **all** should be specified here unless the file system is going to be exported to NFS V4 clients. NFS V4 ACLs affect file attributes (mode) and have access and authorization characteristics that are different from traditional GPFS ACLs.

-K {no | whenpossible | always}

Specifies whether strict replication is to be enforced:

no Strict replication is not enforced. GPFS will try to create the needed number of replicas, but will still return EOK as long as it can allocate at least one replica.

whenpossible

Strict replication is enforced provided the disk configuration allows it. If there is only one failure group, strict replication will not be enforced.

always

Strict replication is enforced.

-m DefaultMetaDataReplicas

Changes the default number of metadata replicas. Valid values are 1 and 2 but cannot exceed the values of *MaxMetaDataReplicas* set when the file system was created.

Changing the default replication settings using the **mmchfs** command does not change the replication setting of existing files. After running the **mmchfs** command, the **mmrestripefs** command with the **-R** option can be used to change *all* existing files or you can use the **mmchattr** command to change a small number of existing files.

-o MountOptions

Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see "GPFS-specific mount options" on page 17.

-Q {yes | no}

If **-Q yes** is specified, quotas are activated automatically when the file system is mounted. If **-Q no** is specified, the quota files remain in the file system, but are not used. Before you activate or deactivate quotas, you must unmount the file system from the cluster.

For additional information, refer to “Enabling and disabling GPFS quota management” on page 41.

-r DefaultDataReplicas

Changes the default number of data replicas. Valid values are 1 and 2 but cannot exceed the values of *MaxDataReplicas* set when the file system was created.

Changing the default replication settings using the **mmchfs** command does not change the replication setting of existing files. After running the **mmchfs** command, the **mmrestripefs** command with the **-R** option can be used to change *all* existing files or you can use the **mmchattr** command to change a small number of existing files.

-S {yes | no}

Suppress the periodic updating of the value of **atime** as reported by the **gpfs_stat()**, **gpfs_fstat()**, **stat()**, and **fstat()** calls. If **yes** is specified these calls report the last time the file was accessed when the file system was mounted with the **-S no** option.

-t DriveLetter

Changes the Windows drive letter for the file system.

The file system must be unmounted on all nodes prior to issuing the command.

-T Mountpoint

Change the mount point of the file system starting at the next mount of the file system.

The file system must be unmounted on all nodes prior to issuing the command.

-V {full | compat}

Changes the file system format to the latest format supported by the currently installed level of GPFS. This *may* cause the file system to become permanently incompatible with earlier releases of GPFS.

Note: The **-V** option cannot be used to make file systems created prior to GPFS 3.2.1.5 available to Windows nodes. Windows nodes can mount only file systems that are created with GPFS 3.2.1.5 or later.

Before issuing the **-V** option, see the *Migration, coexistence and compatibility* topic in the *General Parallel File System: Concepts, Planning, and Installation Guide*. You must ensure that all nodes in the cluster have been migrated to the latest level of GPFS code and that you have successfully run the **mmchconfig release=LATEST** command.

For information about specific file system format and function changes when upgrading to the current release, see Chapter 12, “File system format changes between versions of GPFS,” on page 395.

full Enables all new functionality that requires different on-disk data structures. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that have the file system mounted at the time the command is issued, the **mmchfs** command will fail.

compat

Enables only backwardly compatible format changes. Nodes in remote clusters that were able to mount the file system before will still be able to do so.

-W NewDeviceName

Assign *NewDeviceName* to be the device name for the file system.

-z {yes | no}

Enable or disable DMAPI on the file system. For further information on DMAPI for GPFS, see the *General Parallel File System: Data Management API Guide*.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchfs** command.

You may issue the **mmchfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To change the default replicas for metadata to 2 and the default replicas for data to 2 for new files created in the **fs0** file system, issue this command:

```
mmchfs fs0 -m 2 -r 2
```

To confirm the change, issue this command:

```
mmllsfs fs0 -m -r
```

The system displays information similar to:

flag	value	description
-m	2	Default number of metadata replicas
-r	2	Default number of data replicas

See also

"mmcrfs Command" on page 130

"mmdelfs Command" on page 168

"mmdf Command" on page 176

"mmfsck Command" on page 187

"mmllsfs Command" on page 219

"mmrestripefs Command" on page 268

Location

`/usr/lpp/mmfs/bin`

mmchmgr Command

Assigns a new file system manager node or cluster manager node.

Synopsis

mmchmgr *{Device | -c}* [*Node*]

Description

The **mmchmgr** command assigns a new file system manager node or cluster manager node.

Parameters

Device

The device name of the file system for which the file system manager node is to be changed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

-c Changes the cluster manager node.

Node The target node to be appointed as either the new file system manager or cluster manager node. Target nodes for manager functions must be selected according to use:

- Target nodes for cluster manager function must be specified from the list of quorum nodes
- Target nodes for file system manager function should be specified from the list of manager nodes

If *Node* is not specified, the new manager is selected automatically.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

NONE

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchmgr** command.

You may issue the **mmchmgr** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

Assume the file system manager for the file system **gpfs1** is currently **k164n05**. To migrate the file system manager responsibilities to **k164n06**, issue this command:

```
mmchmgr gpfs1 k164n06
```

The system displays information similar to:

```
GPFS: 6027-628 Sending migrate request to current manager node
      89.116.68.69 (k164n05).
GPFS: 6027-629 Node 89.116.68.69 (k164n05) resigned as manager for gpfs1
GPFS: 6027-630 Node 89.116.68.70 (k164n06) appointed as manager for gpfs1
```

To change the cluster manager node, issue the command:

```
mmchmgr -c c5n107
```

The system displays information similar to:

```
Appointing node 9.114.132.107 (c5n107) as cluster manager
Node 9.114.132.107 (c5n107) has taken over as cluster manager
```

To verify the change, issue the command:

```
mmismgr -c
```

The system displays information similar to:

```
Cluster manager node: 9.114.132.107 (c5n107)
```

See also

“mmismgr Command” on page 223

Location

/usr/lpp/mmfs/bin

mmchnode Command

Changes node attributes.

Synopsis

mmchnode *change-options* **-N**{*Node[,Node...]* | *NodeFile* | *NodeClass*}

Or,

mmchnode **{-S** *Filename* | **--spec-file=***Filename*}

Description

Use the **mmchnode** command to change one or more attributes on a single node or on a set of nodes. If conflicting node designation attributes are specified for a given node, the last value is used. If any of the attributes represent a node-unique value, the **-N** option must resolve to a single node.

Parameters

-N { *Node[,Node...]* | *NodeFile* | *NodeClass*}

Specifies the nodes whose states are to be changed. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

-S *Filename* | **--spec-file=***Filename*

Specifies a file with a detailed description of the changes to be made. Each line represents the changes to an individual node and has the following format:

node-identifier change-options

change-options

A blank-separated list of attribute[=*value*] pairs. The following attributes can be specified:

--admin-interface={*hostname* | *ip_address*}

Specifies the name of the node to be used by GPFS administration commands when communicating between nodes. The admin node name must be specified as an IP address or a hostname that is resolved by the host command to the desired IP address. If the keyword DEFAULT is specified, the admin interface for the node is set to be equal to the daemon interface for the node.

--client

Specifies that the node should not be part of the pool of nodes from which cluster managers, file system managers, and token managers are selected.

--cnfs-disable

Temporarily disables the CNFS functionality of a CNFS member node.

--cnfs-enable

Enables a previously-disabled CNFS member node.

--cnfs-groupid=*groupid*

Specifies a failover recovery group for the node. If the keyword DEFAULT is specified, the CNFS recovery group for the node is set to zero.

For additional information, refer to “Implementing a clustered NFS using GPFS on Linux” in the *GPFS: Advanced Administration Guide*.

--cnfs-interface=*ip_address_list*

A comma-separated list of host names or IP addresses to be used for GPFS cluster NFS serving.

The specified IP addresses can be real or virtual (aliased). These addresses must be configured to be static (not DHCP) and to not start at boot time.

The GPFS daemon interface for the node cannot be a part of the list of CNFS IP addresses.

If the keyword DEFAULT is specified, the CNFS IP address list is removed and the node is no longer considered a member of CNFS.

For additional information, refer to "Implementing a clustered NFS using GPFS on Linux" in the *GPFS: Advanced Administration Guide*.

--daemon-interface={hostname | ip_address}

Specifies the host name or IP address to be used by the GPFS daemons for node-to-node communication. The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address or a name that is resolved by the host command to that original address.

When changing the daemon interface, GPFS must be stopped on all nodes in the cluster. The keyword DEFAULT is not allowed for this attribute.

--manager

Designates the node as part of the pool of nodes from which cluster managers, file system managers, and token managers are selected. Windows nodes cannot be designated as manager nodes.

--nonquorum

Designates the node as a non-quorum node. If two or more quorum nodes are downgraded at the same time, GPFS must be stopped on all nodes in the cluster. GPFS does not have to be stopped if the nodes are downgraded one at a time.

--nosnmp-agent

Stops the SNMP subagent and specifies that the node should no longer serve as an SNMP collector node. For additional information see the topic "GPFS SNMP support" in the *General Parallel File System: Advanced Administration Guide*.

--quorum

Designates the node as a quorum node. Windows nodes cannot be designated as quorum nodes.

--snmp-agent

Designates the node as an SNMP collector node. If the GPFS daemon is active on this node, the SNMP subagent will be started as well. For additional information see the topic "GPFS SNMP support" in the *General Parallel File System: Advanced Administration Guide*.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchnode** command.

You can issue the **mmchnode** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must make certain that:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password and without any extraneous messages.

Examples

To change nodes k145n04 and k145n05 to be both quorum and manager nodes, issue this command:

```
mmchnode --quorum --manager -N k145n04,k145n05
```

The system displays information similar to:

```
Wed May 16 04:50:24 EDT 2007: mmchnode: Processing node k145n04.kgn.ibm.com
Wed May 16 04:50:24 EDT 2007: mmchnode: Processing node k145n05.kgn.ibm.com
mmchnode: Propagating the cluster configuration data to all affected nodes.
This is an asynchronous process.
```

After completion, **mmclscluster** displays information similar to:

```
GPFS cluster information
=====
GPFS cluster name: mynodes.kgn.ibm.com
GPFS cluster id: 680681553700098206
GPFS UID domain: mynodes.kgn.ibm.com
Remote shell command: /usr/bin/rsh
Remote file copy command: /usr/bin/rcp

GPFS cluster configuration servers:
-----
Primary server: k145n04.kgn.ibm.com
Secondary server: k145n06.kgn.ibm.com

Node Daemon node name IP address Admin node name Designation
-----
1 k145n04.kgn.ibm.com 9.114.68.68 k145n04.kgn.ibm.com quorum-manager
2 k145n05.kgn.ibm.com 9.114.68.69 k145n05.kgn.ibm.com quorum-manager
3 k145n06.kgn.ibm.com 9.114.68.70 k145n06.kgn.ibm.com
```

To change nodes k145n04 and k145n05 to be both quorum and manager nodes, and node k45n06 to be a non-quorum node, issue this command:

```
mmchnode -S /tmp/specFile
```

Where the contents of **/tmp/specFile** are:

```
k145n04 --quorum --manager
k145n05 --quorum --manager
k145n06 --nonquorum
```

The system displays information similar to:

```
Wed May 16 05:23:31 EDT 2007: mmchnode: Processing node k145n04
Wed May 16 05:23:32 EDT 2007: mmchnode: Processing node k145n05
Wed May 16 05:23:32 EDT 2007: mmchnode: Processing node k145n06
Verifying GPFS is stopped on all nodes ...
mmchnode: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

And **mmclscluster** displays information similar to:


```
GPFS cluster information
=====
GPFS cluster name: mynodes.kgn.ibm.com
GPFS cluster id: 680681553700098206
GPFS UID domain: mynodes.kgn.ibm.com
Remote shell command: /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

```
GPFS cluster configuration servers:
-----
Primary server: k145n04.kgn.ibm.com
Secondary server: k145n06.kgn.ibm.com
```

	Node	Daemon	node name	IP address	Admin	node name	Designation
1	k145n04.kgn.ibm.com	9.114.68.68	k145n04.kgn.ibm.com	quorum-manager			
2	k145n05.kgn.ibm.com	9.114.68.69	k145n05.kgn.ibm.com	quorum-manager			
3	k145n06.kgn.ibm.com	9.114.68.70	k145n06.kgn.ibm.com				

See also

“mmchconfig Command” on page 92

“mmlscluster Command” on page 209

Location

/usr/lpp/mmfs/bin

mmchnsd Command

Changes Network Shared Disk (NSD) configuration parameters.

Synopsis

```
mmchnsd {"DiskDesc[;DiskDesc...]" | -F DescFile}
```

Description

The **mmchnsd** command serves several purposes. You can use it to:

- Specify a server list for an NSD that does not have one.
- Change the NSD server nodes specified in the server list.
- Delete the server list. The disk must now be SAN-attached to all nodes in the cluster on which the file system will be mounted.

You must follow these rules when changing NSDs:

- You must identify the disks by the NSD names that were given to them by the **mmcrnsd** command.
- You must explicitly specify values for all NSD servers on the list even if you are only changing one of the values.
- The file system that contains the NSD being changed must be unmounted prior to issuing the **mmchnsd** command.
- The NSD must be properly connected to the new nodes prior to issuing the **mmchnsd** command.
- This command cannot be used to change the *DiskUsage* or *FailureGroup* for an NSD. You must issue the **mmchdisk** command to change these.
- To move a disk from one storage pool to another, use the **mmdeildisk** and **mmadddisk** commands.
- You cannot change the name of the NSD.

Parameters

DiskDesc

A descriptor for each NSD to be changed. Each descriptor is separated by a semicolon (;). The entire list must be enclosed in single or double quotation marks.

-F DescFile

Specifies a file containing a list of disk descriptors, one per line.

Each disk descriptor must be specified in the form:

DiskName:ServerList

DiskName

Is the NSD name that was given to the disk by the **mmcrnsd** command.

ServerList

Is a comma-separated list of NSD server nodes. You can specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list.

| When specifying server nodes for your NSDs, the output of the **mmiscluster** command
| lists the host name and IP address combinations recognized by GPFS. The utilization of
| aliased host names not listed in the **mmiscluster** command output may produce
| undesired results.

If you do not define a *ServerList*, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a value for *ServerList*.

To remove the NSD server list, do not specify a value for *ServerList*.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

NONE

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchnsd** command.

You may issue the **mmchnsd** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

| If the disk **gpfs1nsd** is currently defined with **k145n05** as the first server and **k145n07** as the second server, and you want to replace **k145n05** with **k145n09**, issue this command:

| `mmchnsd "gpfs1nsd:k145n09,k145n07"`

| To confirm the changes, issue this command:

| `mmlnsd -d gpfs1nsd`

| The system displays information similar to:

```
| File system  Disk name  NSD servers
| -----
| fs2          gpfs1nsd    k145n09.ppd.pok.ibm.com,k145n07.ppd.pok.ibm.com
```

| See also

“mmchdisk Command” on page 100

“mmcrcluster Command” on page 124

“mmcrnsd Command” on page 137

“mmlnsd Command” on page 227

Location

/usr/lpp/mmfs/bin

mmchpolicy Command

Establishes policy rules for a GPFS file system.

Synopsis

mmchpolicy *Device PolicyFileName* [-t *DescriptiveName*] [-l {**yes** | **test**}]

Description

Use the **mmchpolicy** command to establish the rules for policy-based lifecycle management of the files in a given GPFS file system. Some of the things that can be controlled with the help of policy rules are:

- File placement at creation time
- Replication factors
- Movement of data between storage pools
- File deletion

The **mmapplypolicy** command must be run to move data between storage pools or delete files.

Policy changes take effect immediately on all nodes that have the affected file system mounted. For nodes that do not have the file system mounted, policy changes take effect upon the next mount of the file system.

For information on GPFS policies, see the chapter *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

Specifies the device name of the file system for which policy information is to be established or changed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**. This must be the first parameter.

PolicyFileName

Specifies the name of the file that contains the policy rules. If you specify **DEFAULT**, GPFS replaces the current policy file with a single policy rule that assigns all newly-created files to the **system** storage pool.

Options

-l {yes | test}

Specifies whether to activate the rules in the policy file *PolicyFileName*.

yes The policy rules are validated and immediately activated. This is the default.

test The policy rules are validated, but not installed.

-t DescriptiveName

Specifies an optional descriptive name to be associated with the policy rules. The string must be less than 256 characters in length. If not specified, the descriptive name defaults to the base name portion of the *PolicyFileName* parameter.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmchpolicy** command.

You may issue the **mmchpolicy** command from any node in the GPFS cluster.

Examples

1. This command validates a policy before it is installed:

```
mmchpolicy fs2 policyfile -I test
```

The system displays output similar to:

```
Validated policy 'policyfile': parsed 3 StgPoolRules,  
3 MigratRules
```

2. This command installs a policy:

```
mmchpolicy fs2 policyfile
```

The system displays output similar to:

```
Policy 'policyfile' installed and broadcast to all nodes.  
Validated policy 'policyfile': parsed 3 StgPoolRules,  
3 MigratRules
```

To confirm the change, issue this command:

```
mmlspolicy fs2
```

The system displays output similar to:

```
Policy file for file system '/dev/fs2':  
  Installed by root@k155n11.kgn.ibm.com on Mon Dec 12  
  16:56:31 2005.  
  First line from original file 'policyfile' was:  
  /* This is the policy for the fs2 GPFS file system. */
```

See also

“mmapplypolicy Command” on page 73

“mmlspolicy Command” on page 230

Location

/usr/lpp/mmfs/bin

mmcrcluster Command

Creates a GPFS cluster from a set of nodes.

Synopsis

```
mmcrcluster -N {NodeDesc[,NodeDesc...] | NodeFile} -p PrimaryServer [-s SecondaryServer] [-r RemoteShellCommand] [-R RemoteFileCopyCommand] [-C ClusterName] [-U DomainName] [-A] [-c ConfigFile]
```

Description

Use the **mmcrcluster** command to create a GPFS cluster.

Upon successful completion of the **mmcrcluster** command, the **/var/mmfs/gen/mmsdrfs** and the **/var/mmfs/gen/mmfsNodeData** files are created on each of the nodes in the cluster. Do not delete these files under any circumstances. For further information, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.

You must follow these rules when creating your GPFS cluster:

- While a node may mount file systems from multiple clusters, the node itself may only be added to a single cluster using the **mmcrcluster** or **mmaddnode** command.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and issue the **mmaddnode** command to add those nodes.
- You must designate at least one but not more than seven nodes as quorum nodes. You are strongly advised to designate the cluster configuration servers as quorum nodes. How many quorum nodes altogether you will have depends on whether you intend to use the node quorum with tiebreaker algorithm or the regular node based quorum algorithm. For more details, see the *General Parallel File System: Concepts, Planning, and Installation Guide* and search for *designating quorum nodes*.
- Because the **rsh** and **rcp** commands provided by the Windows SUA environment do not support GPFS, if the cluster will contain Windows nodes, you cannot take the defaults for the **-r** and **-R** options and must specify alternate remote programs (for example, **ssh** and **scp**).

Parameters

-A Specifies that GPFS daemons are to be automatically started when nodes come up. The default is not to start daemons automatically.

-C ClusterName

Specifies a name for the cluster. If the user-provided name contains dots, it is assumed to be a fully qualified domain name. Otherwise, to make the cluster name unique, the domain of the primary configuration server will be appended to the user-provided name.

If the **-C** flag is omitted, the cluster name defaults to the name of the primary GPFS cluster configuration server.

-c ConfigFile

Specifies a file containing GPFS configuration parameters with values different than the documented defaults. A sample file can be found in **/usr/lpp/mmfs/samples/mmfs.cfg.sample**. See the **mmchconfig** command for a detailed description of the different configuration parameters.

The **-c ConfigFile** parameter should be used only by experienced administrators. Use this file to set up only those parameters that appear in the **mmfs.cfg.sample** file. Changes to any other values may be ignored by GPFS. When in doubt, use the **mmchconfig** command instead.

-N *NodeDesc[,NodeDesc...] | NodeFile*

Specifies node descriptors, which provide information about nodes to be added to the cluster.

NodeFile

Specifies a file containing a list of node descriptors (see below), one per line, to be added to the cluster.

NodeDesc[,NodeDesc...]

Specifies the list of nodes and node designations to be added to the GPFS cluster. Node descriptors are defined as:

NodeName:NodeDesignations:AdminNodeName

where:

NodeName

Specifies the host name or IP address of the node for GPFS daemon-to-daemon communication.

The host name or IP address must refer to the communication adapter over which the GPFS daemons communicate. Aliased interfaces are not allowed. Use the original address or a name that is resolved by the **host** command to that original address. You can specify a node using any of these forms:

- Short host name (for example, h135n01)
- Long, fully-qualified, host name (for example, h135n01.ibm.com)
- IP address (for example, 7.111.12.102)

Regardless of which form you use, GPFS will resolve the input to a host name and an IP address and will store these in its configuration files. It is expected that those values will not change while the node belongs to the cluster.

NodeDesignations

An optional, "-" separated list of node roles:

- **manager** | **client** – Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is **client**. Windows nodes cannot be designated as manager nodes.
- **quorum** | **nonquorum** – Indicates whether a node is counted as a quorum node. The default is **nonquorum**. Windows nodes cannot be designated as quorum nodes.

AdminNodeName

Specifies an optional field that consists of a node name to be used by the administration commands to communicate between nodes. If *AdminNodeName* is not specified, the *NodeName* value is used.

You must provide a *NodeDesc* for each node to be added to the GPFS cluster.

-p *PrimaryServer*

Specifies the primary GPFS cluster configuration server node used to store the GPFS configuration data. This node must be a member of the GPFS cluster.

-R *RemoteFileCopy*

Specifies the fully-qualified path name for the remote file copy program to be used by GPFS. The default value is **/usr/bin/rcp**.

The remote copy command must adhere to the same syntax format as the **rcp** command, but may implement an alternate authentication mechanism.

-r *RemoteShellCommand*

Specifies the fully-qualified path name for the remote shell program to be used by GPFS. The default value is **/usr/bin/rsh**.

The remote shell command must adhere to the same syntax format as the **rsh** command, but may implement an alternate authentication mechanism.

-s SecondaryServer

Specifies the secondary GPFS cluster configuration server node used to store the GPFS cluster data. This node must be a member of the GPFS cluster.

It is suggested that you specify a secondary GPFS cluster configuration server to prevent the loss of configuration data in the event your primary GPFS cluster configuration server goes down. When the GPFS daemon starts up, at least one of the two GPFS cluster configuration servers must be accessible.

If your primary GPFS cluster configuration server fails and you have not designated a secondary server, the GPFS cluster configuration files are inaccessible, and any GPFS administration commands that are issued fail. File system mounts or daemon startups also fail if no GPFS cluster configuration server is available.

-U DomainName

Specifies the UID domain name for the cluster.

A detailed description of the GPFS user ID remapping convention is contained in *UID Mapping for GPFS In a Multi-Cluster Environment* at www.ibm.com/servers/eserver/clusters/library/wp_aix_lit.html.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrcluster** command.

You may issue the **mmcrcluster** command from any node in the GPFS cluster.

A properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To create a GPFS cluster made of all of the nodes listed in the file **/u/admin/nodelist**, using node **k164n05** as the primary server, and node **k164n04** as the secondary server, issue:

```
mmcrcluster -N /u/admin/nodelist -p k164n05 -s k164n04
```

where **/u/admin/nodelist** has the these contents:

```
k164n04.kgn.ibm.com:quorum
k164n05.kgn.ibm.com:quorum
k164n06.kgn.ibm.com
```

The output of the command is similar to:

```
Mon Aug  9 22:14:34 EDT 2004: 6027-1664 mmcrcluster: Processing node
                               k164n04.kgn.ibm.com
Mon Aug  9 22:14:38 EDT 2004: 6027-1664 mmcrcluster: Processing node
```



```

k164n05.kgn.ibm.com
Mon Aug  9 22:14:42 EDT 2004: 6027-1664 mmcrcluster: Processing node
k164n06.kgn.ibm.com
mmcrcluster: Command successfully completed
mmcrcluster: 6027-1371 Propagating the changes to all affected.
nodes. This is an asynchronous process.

```

To confirm the creation, issue this command:

```
mmclscluster
```

The system displays information similar to:

GPFS cluster information

=====

```

GPFS cluster name:      k164n05.kgn.ibm.com
GPFS cluster id:       680681562214606028
GPFS UID domain:       k164n05.kgn.ibm.com
Remote shell command:  /usr/bin/rsh
Remote file copy command: /usr/bin/rcp

```

GPFS cluster configuration servers:

```

Primary server:   k164n05.kgn.ibm.com
Secondary server: k164n04.kgn.ibm.com

```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	198.117.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	198.117.68.71	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	198.117.68.70	k164n06.kgn.ibm.com	

See also

“mmaddnode Command” on page 70

“mmchconfig Command” on page 92

“mmdelnode Command” on page 170

“mmclscluster Command” on page 209

“mmclsconfig Command” on page 211

Location

/usr/lpp/mmfs/bin

mmcrfileset Command

Creates a GPFS fileset.

Synopsis

mmcrfileset *Device FilesetName* [-t *Comment*]

Description

The **mmcrfileset** command constructs a new fileset with the specified name. The new fileset is empty except for a root directory, and does not appear in the directory name space until the **mmlinkfileset** command is issued. The **mmcrfileset** command is separate from the **mmlinkfileset** command to allow the administrator to establish policies and quotas on the fileset before it is linked into the name space.

For information on GPFS filesets, see the chapter *Policy-based data management for GPFS in General Parallel File System: Advanced Administration Guide*.

The maximum number of filesets that GPFS supports is 1000 filesets per file system.

Parameters

Device

The device name of the file system to contain the new fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

FilesetName

Specifies the name of the newly created fileset.

-t *Comment*

Specifies an optional comment that appears in the output of the **mmllsfileset** command. This comment must be less than 256 characters in length.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrfileset** command.

You may issue the **mmcrfileset** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This example creates two filesets in file system **gpfs1**:

```
mmcrfileset gpfs1 fset1
```

The system displays output similar to:

```
Fileset 'fset1' created.
```

```
mmcrfileset gpfs1 fset2 -t "another fileset"
```

The system displays output similar to:

```
Fileset 'fset2' created.
```

To confirm the change, issue this command:

```
mm lsfileset gpfs1 -L
```

The system displays output similar to:

Name	Id	RootInode	ParentId	Created	Comment
root	0	3	--	Thu Dec 12 15:59:18 2005	root fileset
fset1	1	261633	0	Mon Dec 13 13:55:28 2005	
fset2	2	261634	1	Mon Dec 13 13:56:12 2005	another fileset

See also

“mmchfileset Command” on page 107

“mmdelfileset Command” on page 165

“mmlinkfileset Command” on page 205

“mm lsfileset Command” on page 216

“mmunlinkfileset Command” on page 288

Location

/usr/lpp/mmfs/bin

mmcrfs Command

Creates a GPFS file system.

Synopsis

```
mmcrfs Device {"DiskDesc;DiskDesc..." | -F DescFile} [-A {yes | no | automount}] [-D {nfs4 | posix}]  
[-B BlockSize] [-E {yes | no}] [-j {cluster | scatter}] [-k {posix | nfs4 | all}] [-K {no | whenpossible |  
always}] [-L LogFileSize] [-m DefaultMetadataReplicas] [-M MaxMetadataReplicas] [-n NumNodes] [-N  
NumInodes[:NumInodesToPreallocate]] [-Q {yes | no}] [-r DefaultDataReplicas] [-R MaxDataReplicas] [-S  
| {yes | no}] [-t DriveLetter] [-T MountPoint] [-v {yes | no}] [-z {yes | no}] [--version Version]
```

Description

Use the **mmcrfs** command to create a GPFS file system. The first two parameters *must* be *Device* and either *DiskDescList* or *DescFile* and they *must* be in that order. The block size and replication factors chosen affect file system performance. A maximum of 256 file systems can be mounted in a GPFS cluster at one time, including remote file systems.

When deciding on the maximum number of files (number of inodes) in a file system, consider that for file systems that will be doing parallel file creates, if the total number of free inodes is not greater than 5% of the total number of inodes, there is the potential for slowdown in file system access. The total number of inodes can be increased using the **mmchfs** command.

When deciding on a block size for a file system, consider these points:

1. Supported block sizes are 16 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, and 4 MB.
2. The GPFS block size determines:
 - The minimum disk space allocation unit. The minimum amount of space that file data can occupy is a sub-block. A sub-block is 1/32 of the block size.
 - The maximum size of a read or write request that GPFS sends to the underlying disk driver.
3. From a performance perspective, it is recommended that you set the GPFS block size to match the application buffer size, the RAID stripe size, or a multiple of the RAID stripe size. If the GPFS block size does not match the RAID stripe size, performance may be severely degraded, especially for write operations.
4. In file systems with a high degree of variance in the size of files within the file system, using a small block size would have a large impact on performance when accessing large files. In this kind of system it is suggested that you use a block size of 256 KB (8 KB sub-block). Even if only 1% of the files are large, the amount of space taken by the large files usually dominates the amount of space used on disk, and the waste in the sub-block used for small files is usually insignificant. For further performance information, see the GPFS white papers at http://www.ibm.com/systems/clusters/library/wp_lit.html.
5. The effect of block size on file system performance largely depends on the application I/O pattern.
 - A larger block size is often beneficial for large sequential read and write workloads.
 - A smaller block size is likely to offer better performance for small file, small random read and write, and metadata-intensive workloads.
6. The efficiency of many algorithms that rely on caching file data in a GPFS page pool depends more on the number of blocks cached rather than the absolute amount of data. For a page pool of a given size, a larger file system block size would mean fewer blocks cached. Therefore, when you create file systems with a block size larger than the default of 256 KB, it is recommended that you increase the page pool size in proportion to the block size.
7. The file system block size must not exceed the value of the GPFS **maxblocksize** configuration parameter. The **maxblocksize** parameter is set to 1 MB by default. If a larger block size is desired, use the **mmchconfig** command to increase the **maxblocksize** before starting GPFS.

Results

Upon successful completion of the **mmcrfs** command, these tasks are completed on all GPFS nodes:

- Mount point directory is created.
- File system is formatted.

Parameters

Device

The device name of the file system to be created.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**. However, file system names must be unique within a GPFS cluster. Do not specify an existing entry in **/dev**.

-D {nfs4 | posix}

Specifies whether a 'deny-write open lock' will block writes, which is expected and required by NFS V4. File systems supporting NFS V4 must have **-D nfs4** set. The option **-D posix** allows NFS writes even in the presence of a deny-write open lock. If you intend to export the file system using NFS V4 or Samba, you must use **-D nfs4**. For NFS V3 (or if the file system is not NFS exported at all) use **-D posix**. The default is **-D nfs4**.

-F DescFile

Specifies a file containing a list of disk descriptors, one per line. You may use the rewritten *DiskDesc* file created by the **mmcrnsd** command, create your own file, or enter the disk descriptors on the command line. When using the *DiskDesc* file created by the **mmcrnsd** command, the values supplied on input to the command for *Disk Usage* and *FailureGroup* are used. When creating your own file or entering the descriptors on the command line, you must specify these values or accept the system defaults.

"DiskDesc[;DiskDesc...]"

A descriptor for each disk to be included. Each descriptor is separated by a semicolon (;). The entire list must be enclosed in quotation marks (' or ").

A disk descriptor is defined as (second, third and sixth fields reserved):

DiskName:::DiskUsage:FailureGroup:::StoragePool:

DiskName

You must specify the name of the NSD previously created by the **mmcrnsd** command. For a list of available disks, issue the **mmlnsd -F** command.

DiskUsage

Specify a disk usage or accept the default:

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly

Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

FailureGroup

A number identifying the failure group to which this disk belongs. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. If you do not specify a failure group, the value defaults to the node number of the first NSD server defined in the NSD server list plus 4000. If you do not specify an NSD server list, the value defaults to -1. GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter should be placed in the same failure group.

If replication of **-m** or **-r** is set to 2, storage pools must have two failure groups for the commands to work properly.

StoragePool

Specifies the storage pool to which the disk is to be assigned. If this name is not provided, the default is **system**.

Only the **system** pool may contain **descOnly**, **metadataOnly** or **dataAndMetadata** disks.

Options

-A {yes | no | automount}

Indicates when the file system is to be mounted:

yes When the GPFS daemon starts. This is the default.

no Manual mount.

automount

On non-Windows nodes, when the file system is first accessed. On Windows nodes, when the GPFS daemon starts.

-B BlockSize

Size of data blocks. Must be 16 KB, 64 KB, 256 KB (the default), 512 KB, 1 MB, 2 MB, or 4 MB. Specify this value with the character **K** or **M**, for example 512K.

-E {yes | no}

Specifies whether to report *exact* **mtime** values (**-E yes**), or to periodically update the **mtime** value for a file system (**-E no**). If it is more desirable to display exact modification times for a file system, specify or use the default **-E yes** option.

-j {cluster | scatter}

Specifies the block allocation map type. When allocating blocks for a given file, GPFS first uses a round-robin algorithm to spread the data across all disks in the file system. After a disk is selected, the location of the data block on the disk is determined by the block allocation map type. If **cluster** is specified, GPFS attempts to allocate blocks in clusters. Blocks that belong to a particular file are kept adjacent to each other within each cluster. If **scatter** is specified, the location of the block is chosen randomly.

The **cluster** allocation method may provide better disk performance for some disk subsystems in relatively small installations. The benefits of clustered block allocation diminish when the number of nodes in the cluster or the number of disks in a file system increases, or when the file system's free space becomes fragmented. The **cluster** allocation method is the default for GPFS clusters with eight or fewer nodes and for files systems with eight or fewer disks.

The **scatter** allocation method provides more consistent file system performance by averaging out performance variations due to block location (for many disk subsystems, the location of the data relative to the disk edge has a substantial effect on performance). This allocation method is appropriate in most cases and is the default for GPFS clusters with more than eight nodes or file systems with more than eight disks.

The block allocation map type cannot be changed after the file system has been created.

-k {posix | nfs4 | all}

Specifies the type of authorization supported by the file system:

posix Traditional GPFS ACLs only (NFS V4 ACLs are not allowed). Authorization controls are unchanged from earlier releases.

nfs4 Support for NFS V4 ACLs only. Users are not allowed to assign traditional GPFS ACLs to any file system objects (directories and individual files).

all Any supported ACL type is permitted. This includes traditional GPFS (**posix**) and NFS V4 ACLs (**nfs4**).

The administrator is allowing a mixture of ACL types. For example, **fileA** may have a **posix** ACL, while **fileB** in the same file system may have an NFS V4 ACL, implying different access characteristics for each file depending on the ACL type that is currently assigned. The default is **-k all**.

Neither **nfs4** nor **all** should be specified here unless the file system is going to be exported to NFS V4 clients. NFS V4 ACLs affect file attributes (mode) and have access and authorization characteristics that are different from traditional GPFS ACLs.

-K {no | whenpossible | always}

Specifies whether strict replication is to be enforced:

no Indicates that strict replication is not enforced. GPFS will try to create the needed number of replicas, but will still return EOK as long as it can allocate at least one replica.

whenpossible

Indicates that strict replication is enforced provided the disk configuration allows it. If the number of failure groups is insufficient, strict replication will not be enforced. This is the default value.

always

Indicates that strict replication is enforced.

-L LogFileSize

Specifies the size of the internal log file. The default size is 4 MB or 32 times the file system block size, whichever is smaller. The minimum size is 256 KB and the maximum size is 32 times the file system block size or 16 MB, whichever is smaller. Specify this value with the K or M character, for example: 8M. This value *cannot* be changed after the file system has been created.

In most cases, allowing the log file size to default works well. Increasing the log file size is useful for sites that have a large amount of metadata activity, such as creating and deleting many small files or performing extensive block allocation and deallocation of large files.

-m DefaultMetadataReplicas

Specifies the default number of copies of inodes, directories, and indirect blocks for a file. Valid values are 1 and 2, but cannot be greater than the value of *MaxMetadataReplicas*. The default is 1.

-M MaxMetadataReplicas

Specifies the default maximum number of copies of inodes, directories, and indirect blocks for a file. Valid values are 1 and 2, but cannot be less than the value of *DefaultMetadataReplicas*. The default is 2.

-n NumNodes

The estimated number of nodes that will mount the file system. This is used as a best guess for the initial size of some file system data structures. The default is 32. This value cannot be changed after the file system has been created.

When you create a GPFS file system, you might want to overestimate the number of nodes that will mount the file system. GPFS uses this information for creating data structures that are essential for achieving maximum parallelism in file system operations (see the topic *GPFS*

architecture in *General Parallel File System: Concepts, Planning, and Installation Guide*). Although a large estimate consumes additional memory, underestimating the data structure allocation can reduce the efficiency of a node when it processes some parallel requests such as the allotment of disk space to a file. If you cannot predict the number of nodes that will mount the file system, allow the default value to be applied. If you are planning to add nodes to your system, you should specify a number larger than the default. However, do not make estimates that are not realistic. Specifying an excessive number of nodes may have an adverse affect on buffer operations.

-N *NumInodes[:NumInodesToPreallocate]*

Specifies the maximum number of files in the file system. This value defaults to the size of the file system at creation divided by 1M and is constrained by the formula:

maximum number of files = (total file system space/2) / (inode size + subblock size)

For file systems that will be creating parallel files, if the total number of free inodes is not greater than 5% of the total number of inodes, file system access might slow down. Take this into consideration when creating your file system.

The parameter *NumInodesToPreallocate* specifies the number of inodes that the system will immediately preallocate. If you do not specify a value for *NumInodesToPreallocate*, GPFS will dynamically allocate inodes as needed.

You can specify the *NumInodes* and *NumInodesToPreallocate* values with a suffix, for example 100K or 2M.

-Q {yes | no}

Activates quotas automatically when the file system is mounted. The default is **-Q no**.

To activate GPFS quota management after the file system has been created:

1. Mount the file system.
2. To establish default quotas:
 - a. Issue the **mmdefedquota** command to establish default quota values.
 - b. Issue the **mmdefquotaon** command to activate default quotas.
3. To activate explicit quotas:
 - a. Issue the **mmmedquota** command to activate quota values.
 - b. Issue the **mmquotaon** command to activate quota enforcement.

-r *DefaultDataReplicas*

Specifies the default number of copies of each data block for a file. Valid values are 1 and 2, but cannot be greater than the value of *MaxDataReplicas*. The default is 1.

-R *MaxDataReplicas*

Specifies the default maximum number of copies of data blocks for a file. Valid values are 1 and 2. The value cannot be less than the value of *DefaultDataReplicas*. The default is 2.

-S {yes | no}

Suppresses the periodic updating of the value of **atime** as reported by the **gpfs_stat()**, **gpfs_fstat()**, **stat()**, and **fstat()** calls. The default value is **-S no**. Specifying **-S yes** for a new file system results in reporting the time the file system was created.

-t *DriveLetter*

Specifies the drive letter to be associated with the file system when mounted in a Windows environment.

-T *MountPoint*

Specifies the mount point directory of the GPFS file system. If it is not specified, the mount point will be set to *DefaultMountDir/Device*. The default value for *DefaultMountDir* is **/gpfs** but, it can be changed with the **mmchconfig** command.

-v {yes | no}

Verifies that specified disks do not belong to an existing file system. The default is **-v yes**. Specify

-v no only when you want to reuse disks that are no longer needed for an existing file system. If the command is interrupted for any reason, you must use the **-v no** option on the next invocation of the command.

-z {yes | no}

Enable or disable DMAPI on the file system. The default is **-z no**. For further information on DMAPI for GPFS, see *General Parallel File System: Data Management API Guide*.

--version Version

Enable only the file system features that are compatible with the specified release. The allowed *Version* values are 3.1.0.0, 3.2.0.0, and 3.2.1.5.

The default is 3.2.1.5, which will enable all currently available features, but will prevent nodes that are running earlier GPFS releases from accessing the file system. Windows nodes can mount only file systems that are created with GPFS 3.2.1.5 or later.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrfs** command.

You may issue the **mmcrfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This example shows how to create a file system named **gpfs1** using three disks, each with a block size of 512 KB, allowing metadata and data replication to be 2, turning quotas on, and creating **gpfs1** as the mount point. To complete this task, issue the command:

```
mmcrfs gpfs1 "hd2n97;hd3n97;hd4n97" -B 512K -M 2 -R 2 -Q yes -T /gpfs1
```

The system displays output similar to:

```
The following disks of gpfs1 will be formatted on
node c5n109.ppd.pok.ibm.com:
  hd2n97: size 142028570 KB
  hd3n97: size 142028570 KB
  hd4n97: size 142028570 KB
Formatting file system ...
Disks up to size 300 GB can be added to storage pool 'system'.
Creating Inode File
  46 % complete on Wed May 16 16:03:06 2007
  92 % complete on Wed May 16 16:03:11 2007
 100 % complete on Wed May 16 16:03:11 2007
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
```

```
50 % complete on Wed May 16 16:03:47 2007
100 % complete on Wed May 16 16:03:52 2007
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

See also

“mmchfs Command” on page 109

“mmdelfs Command” on page 168

“mmdf Command” on page 176

“mmedquota Command” on page 182

“mmfsck Command” on page 187

“mmlsfs Command” on page 219

Location

/usr/lpp/mmfs/bin

mmcrnsd Command

Creates Network Shared Disks (NSDs) used by GPFS.

Synopsis

mmcrnsd -F *DescFile* [-v {**yes** |no}]

Description

The **mmcrnsd** command is used to create cluster-wide names for NSDs used by GPFS.

This is the first GPFS step in preparing a disk for use by a GPFS file system. A disk descriptor file supplied to this command is rewritten with the new NSD names and that rewritten disk descriptor file can then be supplied as input to the **mmcrfs**, **mmadddisk** or **mmrpldisk** commands.

The name created by the **mmcrnsd** command is necessary since disks connected at multiple nodes may have differing disk device names in **/dev** on each node. The name uniquely identifies the disk. This command must be run for all disks that are to be used in GPFS file systems. The **mmcrnsd** command is also used to assign an NSD server list that can be used for I/O operations on behalf of nodes that do not have direct access to the disk.

To identify that the disk has been processed by the **mmcrnsd** command, a unique NSD volume ID is written on sector 2 of the disk. All of the NSD commands (**mmcrnsd**, **mmlsnsd**, and **mmdeinsd**) use this unique NSD volume ID to identify and process NSDs.

After the NSDs are created, the GPFS cluster data is updated and they are available for use by GPFS.

When using an IBM eServer High Performance Switch (HPS) in your configuration, it is suggested you process your disks in two steps:

1. Create virtual shared disks on each physical disk through the **mmcrvsd** command.
2. Using the rewritten disk descriptors from the **mmcrvsd** command, create NSDs through the **mmcrnsd** command.

Results

Upon successful completion of the **mmcrnsd** command, these tasks are completed:

- NSDs are created.
- The *DescFile* contains NSD names to be used as input to the **mmcrfs**, **mmadddisk**, or the **mmrpldisk** commands.
- A unique NSD volume ID to identify the disk as an NSD has been written on sector 2.
- An entry for each new disk is created in the GPFS cluster data.

Parameters

-F *DescFile*

Specifies the file containing the list of disk descriptors, one per line. Disk descriptors have this format:

DiskName:ServerList::DiskUsage:FailureGroup:DesiredName:StoragePool

DiskName

Is the block device name appearing in **/dev** for the disk you want to define as an NSD. Examples of disks that are accessible through a block device are SAN-attached disks or virtual shared disks. If a server nodes is specified, *DiskName* must be the **/dev** name for the disk device of the first listed NSD server node.

See the Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for the latest supported disk types.

GPFS provides the **mmcrvsd** helper command to ease configuration of **/dev** disk devices. In an AIX environment, this command can be used to configure virtual shared disks and make them accessible to nodes connected over a high performance switch. The output disk descriptor file from an **mmcrvsd** command can be used as input to the **mmcrnsd** command, since the virtual shared disk names enumerated in that file will appear as **/dev** block devices on switch attached nodes.

ServerList

Is a comma separated list of NSD server nodes. You may specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list.

When specifying server nodes for your NSDs, the output of the **mmclscluster** command lists the host name and IP address combinations recognized by GPFS. The utilization of aliased host names not listed in the **mmclscluster** command output may produce undesired results.

If you do not define a *ServerList*, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a *ServerList*.

DiskUsage

Specify a disk usage or accept the default. This field is ignored by the **mmcrnsd** command, and is passed unchanged to the output descriptor file produced by the **mmcrnsd** command. Possible values are:

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly

Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

FailureGroup

Is a number identifying the failure group to which this disk belongs. You can specify any value from -1 to 4000 (where -1 indicates that the disk has no point of failure in common with any other disk). If you do not specify a failure group, the value defaults to the node number plus 4000 for the first NSD server defined in the server list. If you do not specify an NSD server list, the value defaults to -1.

GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single failure. All disks that are attached to the same NSD server or adapter should be placed in the same failure group.

DesiredName

Specify the name you desire for the NSD to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string 'gpfs'.

Note: This name can contain only the following characters: 'A' through 'Z', 'a' through 'z', '0' through '9', or '_' (the underscore). All other characters are not valid.

If a desired name is not specified, the NSD is assigned a name according to the convention:

gpfs/NNnsd

where *NN* is a unique nonnegative integer not used in any prior NSD.

StoragePool

Specifies the name of the storage pool that the NSD is assigned to. This field is ignored by the **mmcrnsd** command, and is passed unchanged to the output descriptor file produced by the **mmcrnsd** command.

Upon successful completion of the **mmcrnsd** command, the *DescFile* file is rewritten to contain the created NSD names in place of the device name. NSD servers defined in the *ServerList* and *desiredName* are omitted from the rewritten disk descriptor and all other fields, if specified, are copied without modification. The original lines, as well as descriptor lines in error, are commented out and preserved for reference. The rewritten disk descriptor file can then be used as input to the **mmcrfs**, **mmadddisk**, or the **mmrpldisk** commands. You must have **write** access to the directory where the *DescFile* file is located in order to rewrite the created NSD information.

The *Disk Usage* and *Failure Group* specifications in the disk descriptor are preserved only if you use the rewritten file produced by the **mmcrnsd** command. If you do not use this file, you must either accept the default values or specify new values when creating disk descriptors for other commands.

Options

-v {yes|no}

Verify the disk is not already formatted as an NSD.

A value of **-v yes** specifies that the NSD are to be created only if the disk has not been formatted by a previous invocation of the **mmcrnsd** command, as indicated by the NSD volume ID on sector 2 of the disk. A value of **-v no** specifies that the disk is to be formatted irrespective of its previous state. The default is **-v yes**.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrnsd** command.

You may issue the **mmcrnsd** command from any node in the GPFS cluster.

When using the **rsh** and **rlogin** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory, on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.

2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

The NSD **ServerList** uses the form:

```
server1[,server2,...,server8]
```

To create your NSDs from the descriptor file **nsdesc** containing:

```
sdav1:k145n05,k145n06::dataOnly:4::poolA
sdav2:k145n06,k145n05::dataAndMetadata:5:ABC
```

Issue this command:

```
mmcrnsd -F nsdesc
```

These descriptors translate as:

Disk Name

sdav1

Server List

k145n05,k145n06

Disk Usage

dataOnly

Failure Group

4

Storage Pool

poolA

and

Disk Name

sdav2

Server List

k145n06,k145n05

Disk Usage

dataAndMetadata, allowing both

Failure Group

5

Desired Name

gpfs20nsd

Storage Pool

system

nsdesc is rewritten as

```
#sdav1:k145n05,k145n06::dataOnly:4::poolA
gpfs20nsd::dataOnly:4::poolA
#sdav2:k145n06,k145n05::dataAndMetadata:5:ABC
ABC::dataAndMetadata:5
```

The output is similar to this:

```
mmcrnsd: Processing disk sdav1
mmcrnsd: Processing disk sdav2
mmcrnsd: 6027-1371 Propagating the changes to all affected
          nodes. This is an asynchronous process.
```

See also

“mmadddisk Command” on page 66

“mmcrfs Command” on page 130

“mmdeldisk Command” on page 162

“mmdelnsd Command” on page 172

“mmlsnsd Command” on page 227

“mmrpldisk Command” on page 271

Location

/usr/lpp/mmfs/bin

mmcrsnapshot Command

Creates a snapshot of an entire GPFS file system at a single point in time.

Synopsis

mmcrsnapshot *Device Directory*

Description

Use the **mmcrsnapshot** command to create a snapshot of an entire GPFS file system at a single point in time.

A snapshot is a copy of the changed user data in the file system. System data and existing snapshots are not copied. The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time the copy was created. Snapshots are exact copies of changed data in the active files and directories of a file system. Snapshots of a file system are read-only and they appear in a **.snapshots** directory located in the file system root directory. The files and attributes of the file system may be changed only in the active copy.

There is a maximum limit of 31 snapshots per file system. Snapshots may be deleted only by issuing the **mmdelsnapshot** command. The **.snapshots** directory cannot be deleted, though it can be renamed with the **mmsnapdir** command using the **-s** option.

If the **mmcrsnapshot** command is issued while a conflicting command is running, the **mmcrsnapshot** command waits for that command to complete. If the **mmcrsnapshot** command is running while a conflicting command is issued, the conflicting command waits for the **mmcrsnapshot** command to complete. Conflicting operations include:

1. Other snapshot commands
2. Adding, deleting, replacing disks in the file system
3. Rebalancing, repairing, reducing disk fragmentation in a file system

If quorum is lost before the **mmcrsnapshot** command completes, the snapshot is considered partial and will be deleted when quorum is achieved again.

Because snapshots are not full, independent copies of the entire file system, they should not be used as protection against media failures. For protection against media failures, see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *recoverability considerations*.

For more information on snapshots, see *Creating and maintaining snapshots of GPFS file systems* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system for which the snapshot is to be created. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Directory

Specifies the name given to the snapshot. The name appears as a subdirectory of the **.snapshots** directory in the file system root. Each snapshot must have a unique name.

If you do not want to traverse the file system's root to access the snapshot, a more convenient mechanism that enables a connection in each directory of the active file system can be enabled with the **-a** option of the **mmsnapdir** command.

Options

NONE

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmcrsnapshot** command.

You may issue the **mmcrsnapshot** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To create a snapshot **snap1**, for the file system **fs1**, issue this command:

```
mmcrsnapshot fs1 snap1
```

The output is similar to this:

```
Writing dirty data to disk
Quiescing all file system operations
Writing dirty data to disk again
Creating snapshot.
Resuming operations.
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If a second snapshot were to be created at a later time, the first snapshot would remain as is. Snapshots are made only of active file systems, not existing snapshots. For example:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk
Quiescing all file system operations
Writing dirty data to disk again
Creating snapshot.
Resuming operations.
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

```
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

```
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userA/file2
/fs1/.snapshots/snap2/userA/file3
```

See also

“mmdelsnapshot Command” on page 174

“mmlssnapshot Command” on page 235

“mmrestorefs Command” on page 262

“mmsnapdir Command” on page 277

Location

/usr/lpp/mmfs/bin

mmcrvsd Command

Creates virtual shared disks for use by GPFS.

Synopsis

mmcrvsd [-f *FanoutNumber*] [-y] [-c] -F *DescFile*

Description

The **mmcrvsd** command can be used to create virtual shared disks for subsequent use by the **mmcrnsd** command. IBM Virtual shared disk is a subsystem that permits application programs that are running on different nodes of an RSCT peer domain access a raw logical volume as if it were local at each of the nodes. Virtual shared disks created with **mmcrvsd** follow the convention of one local volume group, one local logical volume, one global volume group, and one virtual shared disk per physical volume. After the virtual shared disk is created, it is configured and started on each node with a defined virtual shared disk adapter. See the **updatevsdnode** command in the correct manual for your environment at <http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html>.

The **mmcrvsd** command can be used only in the AIX environment.

Where possible, the **mmcrvsd** command creates and starts virtual shared disk components in parallel. For instance, when multiple physical disk servers are specified in the disk descriptor file, their LVM components are created in parallel. Starting of all virtual shared disks, on all nodes, always occurs in parallel.

The **mmcrvsd** command may also be restarted should one of the steps fail.

Results

Upon successful completion of the **mmcrvsd** command:

- Virtual shared disks are created.

If a desired name **vsdname** is specified on the disk descriptor, **mmcrvsd** uses that name for the name of the virtual shared disk. If a desired name is not specified, the virtual shared disk is assigned a name according to the convention:

gpfs/NN/vsd

where *NN* is a unique nonnegative integer not used in any prior virtual shared disk named with this convention.

- Virtual shared disks are synchronously started on all nodes.
- If a desired name **vsdname** is specified on the disk descriptor, **mmcrvsd** uses that name as the basis for the names of the global volume group, local logical volume, and local volume group according to the convention:

vsdnamevg

the global volume group

vsdnamelv

the local logical volume

vsdnamevg

the local volume group

- If a desired name is not specified, the global volume group, local logical volume, and local volume group for the virtual shared disk are named according to the convention:

gpfs/NGvg
the global volume group

gpfs/MLv
the local logical volume

gpfs/MLvg
the local volume group

where **gpfs/MLvsd** was the name chosen for the virtual shared disk.

- The primary server is configured and the volume group is varied online there.
- The backup server is configured and the volume group is imported there, but varied off.
- The *DescFile* file is rewritten to contain the created virtual shared disk names in place of any disk descriptors containing physical disk or vpath names. Primary and backup servers are omitted from the rewritten disk descriptor and all other fields, if specified, are copied without modification. The rewritten disk descriptor file can then be used as input to the **mmcrvsd** command.

Error recovery

Each step of the **mmcrvsd** process is enumerated during command execution. For example at step 0, the **mmcrvsd** command prints:

Step 0: Setting up environment

As each step is started, its corresponding number is recorded in the *DescFile* file as a comment at the end. This comment serves as restart information to subsequent invocations of the **mmcrvsd** command. For example at step one, the recorded comment would be:

```
#MMCRVSD_STEP=0
```

Upon failure, appropriate error messages from the failing system component are displayed along with **mmcrvsd** error messages.

After correcting the failing condition and restarting the **mmcrvsd** command with the same descriptor file, the command prompts you to restart at the last failing step. For example, if a prior invocation of **mmcrvsd** failed at step one, the prompt would be:

```
A prior invocation of this command has recorded a partial  
completion in the file (/tmp/DescFile).  
Should we restart at prior failing step(1)?[y]/n=>
```

The default response is **y**; yes restart at the prior failing step.

Parameters

-F *DescFile*

The file containing the list of disk descriptors, one per line, in the form:

```
DiskName:PrimaryServer:BackupServer:DiskUsage:FailureGroup:DesiredName:StoragePool
```

DiskName

The device name of the disk you want to use to create a virtual shared disk. This can be either an hdisk name or a vpath name for an SDD device. GPFS performance and recovery processes function best with one disk per virtual shared disk. If you want to create virtual shared disks with more than one disk, refer to the correct manual for your environment at <http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsctbooks.html>

PrimaryServer

The name of the virtual shared disk server node. This can be in any recognizable form.

BackupServer

The backup server name. This can be specified in any recognizable form or allowed to default to none.

DiskUsage

Specify a disk usage or accept the default (see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *recoverability considerations*). This field is ignored by the **mmcrvsd** command and is passed unchanged to the output descriptor file produced by the **mmcrvsd** command.

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default.

dataOnly

Indicates that the disk contains data and does not contain metadata.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

Disk usage considerations:

1. The *DiskUsage* parameter is not utilized by the **mmcrvsd** command but is copied intact to the output file that the command produces. The output file may then be used as input to the **mmcrnsd** command.
2. RAID devices are not well-suited for performing small block writes. Since GPFS metadata writes are often smaller than a full block, you may find using non-RAID devices for GPFS metadata better for performance.

FailureGroup

A number identifying the failure group to which this disk belongs. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. All disks that have a common point of failure, such as all disks that are attached to the same virtual shared disk server node, should be placed in the same failure group. The value is passed unchanged to the output descriptor file produced by the **mmcrvsd** command.

DesiredName

Specify the name you desire for the virtual shared disk to be created. This name must not already be used as another GPFS or AIX disk name, and it must not begin with the reserved string 'gpfs'.

Note: This name can contain only the following characters: 'A' through 'Z', 'a' through 'z', '0' through '9', or '_' (the underscore). All other characters are not valid. The maximum size of this name is 13 characters.

StoragePool

Specifies the name of the storage pool that the NSD is assigned to. This field is ignored by the **mmcrvsd** command, and is passed unchanged to the output descriptor file produced by the **mmcrvsd** command.

Options

-f *FanoutNumber*

The maximum number of concurrent nodes to communicate with during parallel operations. The default value is 10.

-y Specifies no prompting for any queries the command may produce. All default values are accepted.

-c Specifies to create Concurrent Virtual Shared Disks. This option is valid only for disk descriptors that specify both a primary and a backup virtual shared disk server.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrvsd** command.

You may issue the **mmcrvsd** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory, on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To create a virtual shared disk with the descriptor file **vsdesc** containing:

```
hdisk2:k145n01:k145n02:dataOnly:4
hdisk3:k145n06::dataAndMetadata:5:ABC
```

These descriptors translate as:

Disk Name

hdisk2

Server Name

k145n01

Backup Server Name

k145n02

Disk Usage

dataOnly

Failure Group

4

Desired Name

Name defaults to gpfs20vsd

and

Disk Name

hdisk3

Server Name

k145n06

Backup Server Name

none

Disk Usage

dataAndMetadata

Failure Group

5

Desired Name

ABC

The low level components of the virtual shared disk **gpfs20vsd** are created:

gpfs20gvg

global volume group

gpfs20lv

local logical volume

gpfs20vg

local volume group

The low level components of the virtual shared disk **ABC** are created:

ABCgvg

global volume group

ABClv local logical volume

ABCvg

local volume group

See also

“mmcrnsd Command” on page 137

Location

/usr/lpp/mmfs/bin

mmdefedquota Command

Sets default quota limits to a file system.

Synopsis

mmdefedquota {-u | -g | -j} *Device*

Description

Use the **mmdefedquota** command to set or change default quota limits for new users, groups, and filesets for a file system. Default quota limits for a file system may be set or changed only if the file system was created with the **-Q yes** option on the **mmcrfs** command or changed with the **mmchfs** command.

The **mmdefedquota** command displays the current values for these limits, if any, and prompts you to enter new values using your default editor:

- Current block usage (display only)
- Current inode usage (display only)
- Inode soft limit
- Inode hard limit
- Block soft limit

Displayed in **KB**, but may be specified using **k, K, m, M, g, or G**. If no suffix is provided, the number is assumed to be in **bytes**.

- Block hard limit

Displayed in **KB**, but may be specified using **k, K, m, M, g, or G**. If no suffix is provided, the number is assumed to be in **bytes**.

Note: A block or inode limit of 0 indicates no limit.

The **mmdefedquota** command waits for the edit window to be closed before checking and applying new values. If an incorrect entry is made, you must reissue the command and enter the correct values.

When setting quota limits for a file system, replication within the file system should be considered. GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmfsquota** command and the **mmrepquota** command are double the value reported by the **ls** command.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

| This command cannot be run from a Windows node.

Parameters

Device

The device name of the file system to have default quota values set for.

File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- g** Specifies that the default quota value is to be applied for new groups accessing the specified file system.
- j** Specifies that the default quota value is to be applied for new filesets in the specified file system.

-u Specifies that the default quota value is to be applied for new users accessing the specified file system.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdefedquota** command.

GPFS must be running on the node from which the **mmdefedquota** command is issued.

Examples

To set default quotas for new users of the file system **fs1**, issue this command:

```
mmdefedquota -u fs1
```

The system displays information in your default editor similar to:

```
*** Edit quota limits for USR DEFAULT entry
NOTE: block limits will be rounded up to the next multiple
      of the block size.
block units may be: K, M, or G.
fs1: blocks in use: 0K, limits (soft = 0K, hard = 0K)
inodes in use: 0, limits (soft = 0, hard = 0)
```

To confirm the change, issue this command:

```
mmfsquota -d -u
```

The system displays information similar to:

		Block Limits					File Limits				
Filesystem	type	KB	quota	limit	in_doubt	grace	files	quota	limit	in_doubt	grace
fs0	USR	0K	2500	10M	0	none	0	100	1000	0	none
fs1	USR	no default limits									
fs2	USR	no default limits									

See also

“mmcheckquota Command” on page 104

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmfsquota Command” on page 232

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmdefquotaoff Command

Deactivates default quota limit usage for a file system.

Synopsis

```
mmdefquotaoff [-u] [-g] [-j] [-v] {Device [Device...] | -a}
```

Description

The **mmdefquotaoff** command deactivates default quota limits for file systems. If default quota limits are deactivated, new users or groups for that file system will then have a default quota limit of 0, indicating no limit.

If neither the **-u**, **-j** or the **-g** option is specified, the **mmdefquotaoff** command deactivates all default quotas.

If the **-a** option is not used, *Device* must be the last parameter specified.

Parameters

Device

The device name of the file system to have default quota values deactivated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- a** Deactivates default quotas for all GPFS file systems in the cluster. When used in combination with the **-g** option, only group quotas are deactivated. When used in combination with the **-u** or **-j** options, only user or fileset quotas, respectively, are deactivated.
- g** Specifies that default quotas for groups are to be deactivated.
- j** Specifies that default quotas for filesets are to be deactivated.
- u** Specifies that default quotas for users are to be deactivated.
- v** Prints a message for each file system in which default quotas are deactivated.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdefquotaoff** command.

GPFS must be running on the node from which the **mmdefquotaoff** command is issued.

Examples

1. To deactivate default user quotas on file system **fs0**, issue this command:

```
mmdefquotaoff -u fs0
```

To confirm the change, issue this command:

```
mmquota -d -u
```

The system displays information similar to:

		Block Limits				File Limits				
Filesystem	type	KB quota	limit	in_doubt	grace	files	quota	limit	in_doubt	grace
fs0	USR	no default limits								

2. To deactivate default group quotas on all file systems, issue this command:

```
mmdefquotaoff -g -a
```

To confirm the change, issue this command:

```
mmfquota -d -g
```

The system displays information similar to:

Default Block Limits		Default File Limits	
Filesystem	type quota limit	quota limit	
fs0:	GRP no default limits		
fs1:	GRP no default limits		
fs2:	GRP no default limits		

See also

“mmcheckquota Command” on page 104

“mmdefquota Command” on page 150

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmfquota Command” on page 232

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmdefquotaon Command

Activates default quota limit usage for a file system.

Synopsis

mmdefquotaon [-u] [-g] [-j] [-v] [-d] {*Device* [*Device...*] | -a}

Description

The **mmdefquotaon** command activates default quota limits for a file system. If default quota limits are not applied, new users, groups, or filesets for that file system will have a quota limit of 0, indicating no limit.

To use default quotas, the file system must have been created or changed with the **-Q yes** option. See the **mmcrfs** and **mmchfs** commands.

If neither the **-u**, **-j** or the **-g** option is specified, the **mmdefquotaon** command activates all default quota limits.

If the **-a** option is not used, *Device* must be the last parameter specified.

Default quotas are established for new users, groups of users or filesets by issuing the **mmdefquota** command. Under the **-d** option, all users without an explicitly set quota limit will have a default quota limit assigned.

Parameters

Device

The device name of the file system to have default quota values activated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- a** Activates default quotas for all GPFS file systems in the cluster. When used in combination with the **-g** option, only group quotas are activated. When used in combination with the **-u** or **-j** options, only user or fileset quotas, respectively, are activated.
- d** Specifies that existing users, groups of users, or filesets with no established quota limits will have default quota values assigned when the **mmdefquota** command is issued.
If this option is not chosen, existing quota entries remain in effect and are not governed by the default quota rules.
- g** Specifies that only a default quota value for group quotas is to be activated.
- j** Specifies that only a default quota value for fileset quotas is to be activated.
- u** Specifies that only a default quota value for users is to be activated.
- v** Prints a message for each file system in which default quotas are activated.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdefquotaon** command.

GPFS must be running on the node from which the **mmdefquotaon** command is issued.

Examples

1. To activate default user quotas on file system **fs0**, issue this command:

```
mmdefquotaon -u fs0
```

To confirm the change, issue this command:

```
mmllsfs fs0 -Q
```

The system displays information similar to:

flag	value	description
-Q	user	Quotas enforced
	user	Default quotas enabled

2. To activate default group quotas on all file systems in the cluster, issue this command:

```
mmdefquotaon -g -a
```

To confirm the change, individually for each file system, issue this command:

```
mmllsfs fs1 -Q
```

The system displays information similar to:

flag	value	description
-Q	group	Quotas enforced
	group	Default quotas enabled

3. To activate user, group and fileset default quotas on file system **fs2**, issue this command:

```
mmdefquotaon fs2
```

To confirm the change, issue this command:

```
mmllsfs fs2 -Q
```

The system displays information similar to:

flag	value	description
-Q	user;group;fileset	Quotas enforced
	user;group;fileset	Default quotas enabled

See also

“mmcheckquota Command” on page 104

“mmchfs Command” on page 109

“mmcrfs Command” on page 130

“mmdefedquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmedquota Command” on page 182

“mmisquota Command” on page 232

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmdefragfs Command

Reduces disk fragmentation by increasing the number of full free blocks available to the file system.

Synopsis

mmdefragfs *Device* [-i] [-u *BlkUtilPct*] [-P *PoolName*] [-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}]

Description

Use the **mmdefragfs** command to reduce fragmentation of a file system. The **mmdefragfs** command moves existing file system data within a disk to make more efficient use of disk blocks. The data is migrated to unused sub-blocks in partially allocated blocks, thereby increasing the number of free full blocks.

The **mmdefragfs** command can be run against a mounted or unmounted file system. However, best results are achieved when the file system is unmounted. When a file system is mounted, allocation status may change causing retries to find a suitable unused sub-block.

Note: On a file system that has a very low level of fragmentation, negative numbers can be seen in the output of **mmdefragfs** for free sub-blocks. This indicates that the block usage has in fact increased after running the **mmdefragfs** command. If negative numbers are seen, it does not indicate a problem and you do not need to rerun the **mmdefragfs** command.

Parameters

Device

The device name of the file system to have fragmentation reduced. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

-P *PoolName*

Specifies the pool name to use.

-N {*Node[,Node...]* | *NodeFile* | *NodeClass* }

Specifies the nodes that can be used in this disk defragmentation. This parameter supports all defined node classes. The default is **all**, which means that all nodes in the GPFS cluster will participate in the disk defragmentation.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

-i Specifies to query the current disk fragmentation state of the file system. Does not perform the actual defragmentation of the disks in the file system.

-u *BlkUtilPct*

The average block utilization goal for the disks in the file system. The **mmdefragfs** command reduces the number of allocated blocks by increasing the percent utilization of the remaining blocks. The command automatically goes through multiple iterations until *BlkUtilPct* is achieved on all of the disks in the file system or until no progress is made in achieving *BlkUtilPct* from one iteration to the next, at which point it exits.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdefragfs** command.

You may issue the **mmdefragfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To query the fragmentation state of file system **fs0**, issue this command:

```
mmdefragfs fs0 -i
```

The system displays information similar to:

disk name	disk size in nSubblk	free subblk		%	%
		in full blocks	subblk in fragments		
gpfs68nsd	4390912	4270112	551	97.249	99.544
gpfs69nsd	4390912	4271360	490	97.277	99.590
(total)	8781824	8541472	1041		99.567

2. To reduce fragmentation of the file system **fs0** on all defined, accessible disks that are not stopped or suspended, issue this command:

```
mmdefragfs fs0
```

The system displays information similar to:

disk name	free subblk		free		%		%	
	in full blocks	blk	subblk in fragments	blk	free	blk	blk util	blk util
	before	after freed	before	after	before	after	before	after
gpfs57nsd	28896	29888	31	1462	463	50.39	52.12	94.86
gpfs60nsd	41728	43200	46	1834	362	59.49	61.59	93.55
(total)	70624	73088	77	3296	825			93.63

3. To reduce fragmentation of all files in the **fs1** file system until the disks have 100% full block utilization, issue this command:

```
mmdefragfs fs1 -u 100
```

The system displays information similar to:

Defragmenting file system 'fs1'...

Defragmenting until full block utilization is 100.00%, currently 99.96%

```
1 % complete on Wed May 16 15:21:49 2007
6 % complete on Wed May 16 15:21:53 2007
12 % complete on Wed May 16 15:21:56 2007
21 % complete on Wed May 16 15:21:59 2007
32 % complete on Wed May 16 15:22:02 2007
47 % complete on Wed May 16 15:22:05 2007
68 % complete on Wed May 16 15:22:08 2007
91 % complete on Wed May 16 15:22:11 2007
```


95 % complete on Wed May 16 15:22:30 2007
 100 % complete on Wed May 16 15:22:38 2007

disk name	free subblk in full			free subblk in		% free blk		% blk util	
	blocks		blk freed	fragments		before	after	before	after
	before	after		before	after				
hd16vsdn10	2192320	2192320	0	216	216	98.57	98.57	99.99	99.99
hd3vsdn01	1082272	1082272	0	200	200	97.50	97.50	99.98	99.98
hd4vsdn01	1077056	1077056	0	173	173	97.03	97.03	99.98	99.98
hd20vsdn02	1082496	1082496	0	400	400	97.52	97.52	99.96	99.96
hd6vsdn01	1077120	1077120	0	120	120	97.04	97.04	99.99	99.99
hd7vsdn01	1077344	1077344	0	246	246	97.06	97.06	99.98	99.98
hd2vsdn01	1084032	1084032	0	336	336	97.66	97.66	99.97	99.97
hd9vsdn01	1078272	1078272	0	217	217	97.14	97.14	99.98	99.98
hd10vsdn01	1080000	1080000	0	263	263	97.30	97.30	99.98	99.98
hd11vsdn02	1109216	1109216	0	110	110	99.93	99.93	99.99	99.99
hd18vsdn10	2196544	2196544	0	306	306	98.76	98.76	99.99	99.99
hd18vsdn02	1083168	1083168	0	246	246	97.58	97.58	99.98	99.98
hd24n01	1079616	1079616	0	200	200	97.07	97.07	99.98	99.98
hd25n01	1079808	1079808	0	229	229	97.08	97.08	99.98	99.98
hd26n01	1085056	1085056	0	205	205	97.56	97.56	99.98	99.98
hd27n01	1085408	1085408	0	237	237	97.59	97.59	99.98	99.98
hd28n01	1085312	1085312	0	193	193	97.58	97.58	99.98	99.98
hd29n01	1085792	1085792	0	260	260	97.62	97.62	99.98	99.98
hd30n01	1111936	1111936	0	53	53	99.97	99.97	100.00	100.00
hd31n01	1111936	1111936	0	53	53	99.97	99.97	100.00	100.00
hd21n01	1078912	1078912	0	292	292	97.00	97.00	99.97	99.97
hd23n01	1079392	1079392	0	249	249	97.05	97.05	99.98	99.98
hd15vsdn10	2192448	2192448	0	208	208	98.58	98.58	99.99	99.99
hd13vsdn02	1082944	1082944	0	222	222	97.56	97.56	99.98	99.98
hd8vsdn01	1082400	1082400	0	242	242	97.51	97.51	99.98	99.98
hd5vsdn01	1081856	1081856	0	216	216	97.46	97.46	99.98	99.98
hd33n09	2192608	2192608	0	280	280	98.57	98.57	99.99	99.99
(total)	33735264	33735264	0	5972	5972			99.98	99.98

Defragmentation complete, full block utilization is 99.96%.
 Re-issue command to try to reach target utilization of 100.00%.

See also

“mmdf Command” on page 176

Location

/usr/lpp/mmfs/bin

mmdelacl Command

Deletes a GPFS access control list.

Synopsis

mmdelacl [-d] *Filename*

Description

Use the **mmdelacl** command to delete the extended entries of an access ACL of a file or directory, or to delete the default ACL of a directory.

| This command cannot be run from a Windows node.

Parameters

Filename

The path name of the file or directory for which the ACL is to be deleted. If the **-d** option is specified, *Filename* must contain the name of a directory.

Options

-d Specifies that the default ACL of a directory is to be deleted.
Since there can be only one NFS V4 ACL (no separate default), specifying the **-d** flag for a file with an NFS V4 ACL is an error. Deleting an NFS V4 ACL necessarily removes both the ACL and any inheritable entries contained in it.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

The **mmdelacl** command may be issued only by the file or directory owner, the root user, or by someone with control (c) authority in the ACL for the file.

You may issue the **mmdelacl** command only from a node in the GPFS cluster where the file system is mounted.

Examples

To delete the default ACL for a directory named **project2**, issue this command:

```
mmdelacl -d project2
```

To confirm the deletion, issue this command:

```
mmgetacl -d project2
```

The system displays information similar to:

```
#owner:uno  
#group:system
```

See also

“mmeditacI Command” on page 179

“mmgetacI Command” on page 196

“mmputacI Command” on page 246

Location

/usr/lpp/mmfs/bin

mmdeldisk Command

Deletes disks from a GPFS file system.

Synopsis

mmdeldisk *Device* {"*DiskName*;*DiskName*..." | **-F** *DescFile*} [**-a**] [**-c**] [**-r**] [**-N** {*Node*,*Node*...} | *NodeFile* | *NodeClass*}]

Description

The **mmdeldisk** command migrates all data that would otherwise be lost to the remaining disks in the file system. It then removes the disks from the file system descriptor and optionally rebalances the file system after removing the disks.

Run the **mmdeldisk** command when system demand is low.

If a replacement for a failing disk is available, use the **mmrpldisk** command in order to keep the file system balanced. Otherwise, use one of these procedures to delete a disk:

- If the disk is not failing and GPFS can still read from it:
 1. Suspend the disk
 2. Restripe to rebalance all data onto other disks
 3. Delete the disk
- If the disk is permanently damaged and the file system is replicated:
 1. Suspend and stop the disk.
 2. restripe and restore replication for the file system, if possible.
 3. Delete the disk from the file system.
- If the disk is permanently damaged and the file system is not replicated, or if the **mmdeldisk** command repeatedly fails, see the *General Parallel File System: Problem Determination Guide* and search for *Disk media failure*.

If the last disk in a storage pool is deleted, the storage pool is deleted. The **mmdeldisk** command is not permitted to delete the **system** storage pool. A storage pool must be empty in order for it to be deleted.

Results

Upon successful completion of the **mmdeldisk** command, these tasks are completed:

- Data that has not been replicated from the target disks is migrated to other disks in the file system.
- Remaining disks are rebalanced, if specified.

Parameters

Device

The device name of the file system to delete the disks from. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**. This must be the first parameter.

"DiskName[;DiskName...]"

Specifies the names of the disks to be deleted from the file system. If there is more than one disk to be deleted, delimit each name with a semicolon (;) and enclose the list in quotation marks.

-F *DiskFile*

Specifies a file that contains the names of the disks (one name per line), to be deleted from the GPFS cluster.

-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass* }

Specifies the nodes that participate in the restripe of the file system after the specified disks have been removed. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the restripe of the file system).

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

- a** Specifies that the **mmdeldisk** command *not* wait for rebalancing to complete before returning. When this flag is specified, the **mmdeldisk** command runs asynchronously and returns after the file system descriptor is updated and the rebalancing scan is started, but it does not wait for rebalancing to finish. If no rebalancing is requested (**-r** option is not specified), this option has no effect.
- c** Specifies that processing continues even in the event that unreadable data exists on the disks being deleted. Data that has not been replicated is lost. Replicated data is not lost as long as the disks containing the replication are accessible.
- r** Rebalance all existing files in the file system to make more efficient use of the remaining disks.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdeldisk** command.

You may issue the **mmdeldisk** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To delete **gpfs2nsd** and **gpfs3nsd** from file system **fs0** and rebalance the files across the remaining disks, issue this command:

```
mmdeldisk fs0 "gpfs2nsd;gpfs3nsd" -r
```

The system displays information similar to:

```
Deleting disks ...
Scanning 'system' storage pool
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
```

```
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
  2 % complete on Tue Feb 14 14:45:18 2006
  3 % complete on Tue Feb 14 14:45:21 2006
  9 % complete on Tue Feb 14 14:45:24 2006
 18 % complete on Tue Feb 14 14:45:27 2006
 27 % complete on Tue Feb 14 14:45:35 2006
 28 % complete on Tue Feb 14 14:45:39 2006
 29 % complete on Tue Feb 14 14:45:43 2006
 30 % complete on Tue Feb 14 14:45:52 2006
 34 % complete on Tue Feb 14 14:46:04 2006
 37 % complete on Tue Feb 14 14:46:18 2006
 45 % complete on Tue Feb 14 14:46:22 2006
 51 % complete on Tue Feb 14 14:46:26 2006
 94 % complete on Tue Feb 14 14:46:29 2006
100 % complete on Tue Feb 14 14:46:32 2006
Scan completed successfully.
tsdelldisk64 completed.
mmdelldisk: Propagating the cluster configuration data to all
  affected nodes. This is an asynchronous process.
```

See also

“mmaddddisk Command” on page 66

“mmchdisk Command” on page 100

“mmlsdisk Command” on page 213

“mmrpldisk Command” on page 271

Location

/usr/lpp/mmfs/bin

mmdelfileset Command

Deletes a GPFS fileset.

Synopsis

mmdelfileset *Device FilesetName* [-f]

Description

The **mmdelfileset** command deletes a GPFS fileset. The **mmdelfileset** command fails if the fileset is currently linked into the name space. By default, the **mmdelfileset** command fails if the fileset contains any contents except for an empty root directory.

The root fileset cannot be deleted.

If the deleted fileset is included in a snapshot, the fileset is deleted from the active file system, but remains part of the file system in a deleted state. Filesets in the deleted state are displayed by the **mmlsfileset** command with their names in parenthesis. If the **-L** flag is specified, the latest including snapshot is also displayed. A deleted fileset's contents are still available in the snapshot (that is, through some path name containing a **.snapshots** component), since it was saved when the snapshot was created. **mmlsfileset** command illustrates the display of a deleted fileset. When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

The delete operation fails if fileset being deleted is not empty. You need to specify **-f** option to delete a non-empty fileset. When **-f** is specified, all of a fileset's child filesets are unlinked, but their content is unaffected.

For information on GPFS filesets, see the topic *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system that contains the fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

FilesetName

Specifies the name of the fileset to be deleted.

Options

-f Forces the deletion of the fileset. All fileset contents are deleted. Any child filesets are first unlinked.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdelfileset** command.

You may issue the **mmdelfileset** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. This sequence of commands illustrates what happens when attempting to delete a fileset that is linked.

a. `mmfsfileset gpfs1`

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
fset1     Linked  /gpfs1/fset1
fset2     Unlinked --
```

b. `mmdelfileset gpfs1 fset1`

The system displays output similar to:

Fileset fset1 must be unlinked to be deleted.

c. `mmdelfileset gpfs1 fset2`

The system displays output similar to:

Fileset 'fset2' deleted.

d. To confirm the change, issue this command:

`mmfsfileset gpfs1`

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
fset1     Linked  /gpfs1/fset1
```

2. This sequence of commands illustrates what happens when attempting to delete a fileset that contains user files.

a. `mmfsfileset gpfs1`

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
fset1     Linked  /gpfs1/fset1
fset2     Unlinked --
```

b. `mmdelfileset gpfs1 fset2`

The system displays output similar to:

Fileset 'fset2' contains user files,
but can be deleted with the "-f" option.

c. `mmdelfileset gpfs1 fset2 -f`

The system displays output similar to:

```
Deleting user files ...
83 % complete on Tue Dec 17 11:38:25 2005
86 % complete on Tue Dec 17 11:42:47 2005
88 % complete on Tue Dec 17 11:43:13 2005
91 % complete on Tue Dec 17 11:44:15 2005
94 % complete on Tue Dec 17 11:45:20 2005
97 % complete on Tue Dec 17 11:50:14 2005
100 % complete on Tue Dec 17 11:50:47 2005
Fileset 'fset2' deleted.
```


- d. To confirm the change, issue this command:

```
mmfsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':  
Name           Status    Path  
fset1          Linked   /gpfs1/fset1
```

See also

“mmchfileset Command” on page 107

“mmcrfileset Command” on page 128

“mmlinkfileset Command” on page 205

“mmfsfileset Command” on page 216

“mmunlinkfileset Command” on page 288

Location

/usr/lpp/mmfs/bin

mmdelfs Command

Removes a GPFS file system.

Synopsis

mmdelfs *Device* [-p]

Description

The **mmdelfs** command removes all the structures for the specified file system from the nodes in the cluster.

Before you can delete a file system using the **mmdelfs** command, you must unmount it on all nodes.

Results

Upon successful completion of the **mmdelfs** command, these tasks are completed on all nodes:

- Deletes the character device entry from **/dev**.
- Removes the mount point directory where the file system had been mounted.

Parameters

Device

The device name of the file system to be removed. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

Options

- p** Indicates that the disks are permanently damaged and the file system information should be removed from the GPFS cluster data even if the disks cannot be marked as **available**.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdelfs** command.

You may issue the **mmdelfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To delete file system **fs0**, issue this command:

```
mmdelfs fs0
```

The system displays information similar to:

```
mmdelfs: 6027-1366 Marking the disks as available
GPFS: 6027-573 All data on following disks of fs0 will be destroyed:
    gpfs9nsd
    gpfs10nsd
    gpfs15nsd
    gpfs17nsd
GPFS: 6027-574 Completed deletion of file system fs0.
mmdelfs: 6027-1371 Propagating the changes to all affected
                    nodes. This is an asynchronous process.
```

See also

“mmcrfs Command” on page 130

“mmchfs Command” on page 109

“mmlsfs Command” on page 219

Location

/usr/lpp/mmfs/bin

mmdeinode Command

Removes one or more nodes from a GPFS cluster.

Synopsis

mmdeinode {-a | -N *Node[,Node...]* | *NodeFile* | *NodeClass*}

Description

Use the **mmdeinode** command to delete one or more nodes from the GPFS cluster. You may issue the **mmdeinode** command on any GPFS node.

You must follow these rules when deleting nodes:

1. The node being deleted cannot be the primary or secondary GPFS cluster configuration server unless you intend to delete the entire cluster. Verify this by issuing the **mmiscluster** command. If a node to be deleted is one of the servers and you intend to keep the cluster, issue the **mmchcluster** command to assign another node as the server before deleting the node.
2. A node being deleted cannot be defined as an NSD server for any disk unless you intend to delete the entire cluster. Verify this by issuing the **mmisnsd** command. If a node to be deleted is an NSD server for one or more disks, use the **mmchnsd** command to assign another node as an NSD server for the affected disks.
3. Unless all nodes in the cluster are being deleted, run the **mmdeinode** command from a node that will remain in the cluster.
4. Before you can delete a node, you must unmount all of the GPFS file systems and stop GPFS on the node to be deleted.
5. Exercise caution when shutting down GPFS on quorum nodes. If the number of remaining quorum nodes falls below the requirement for a quorum, you will be unable to perform file system operations. See the *General Parallel File System: Concepts, Planning, and Installation Guide* and search for *quorum*.

Note: Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that if you permanently delete nodes that are being used as contact nodes by other GPFS clusters that can mount your file systems, you should notify the administrators of those GPFS clusters so that they can update their own environments.

Results

Upon successful completion of the **mmdeinode** command, the specified nodes are deleted from the GPFS cluster.

Parameters

-a Delete all nodes in the cluster.

-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}

Specifies the set of nodes to be deleted from the cluster. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

Options

None.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdelnode** command.

You may issue the **mmdelnode** command from any node that will remain in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To delete all of the nodes in the cluster, issue this command:

```
mmdelnode -a
```

The system displays information similar to:

```
Verifying GPFS is stopped on all affected nodes ...  
mmdelnode: Command successfully completed  
mmdelnode: 6027-1371 Propagating the cluster configuration data to  
all affected nodes. This is an asynchronous process.
```

2. To delete nodes **k145n12**, **k145n13**, and **k145n14**, issue this command:

```
mmdelnode -N k145n12,k145n13,k145n14
```

The system displays information similar to:

```
Verifying GPFS is stopped on all affected nodes ...  
mmdelnode: Command successfully completed  
mmdelnode: 6027-1371 Propagating the cluster configuration data to  
all affected nodes. This is an asynchronous process.
```

See also

"mmaddnode Command" on page 70

"mmcrcluster Command" on page 124

"mmchconfig Command" on page 92

"mmlsfs Command" on page 219

"mmlscluster Command" on page 209

Location

/usr/lpp/mmfs/bin

mmdelnsd Command

Deletes Network Shared Disks (NSDs) from the GPFS cluster.

Synopsis

```
mmdelnsd {"DiskName[:DiskName...]" | -F DiskFile}
```

Or,

```
mmdelnsd -p NSDId [-N {Node[,Node] }]
```

Description

The **mmdelnsd** command serves two purposes:

1. Delete NSDs from the GPFS cluster.
2. Remove the unique NSD volume ID left on the disk after the failure of a previous invocation of the **mmdelnsd** command. The NSD had been successfully deleted from the GPFS cluster but there was a failure to clear sector 2 of the disk.

The NSD being deleted cannot be part of any file system. Either the **mmdeldisk** or **mmdelfs** command must be issued prior to deleting the NSD from the GPFS cluster.

The NSD being deleted cannot be a tiebreaker disk. Use the **mmchconfig** command to assign new tiebreaker disks prior to deleting the NSD from the cluster. For information on tiebreaker disks, see quorum

Results

Upon successful completion of the **mmdelnsd** command, these tasks are completed:

- All references to the disk are removed from the GPFS cluster data.
- Sector 2 of the disk is cleared of the unique NSD volume ID.

Parameters

DiskName[:DiskName...]

Specifies the names of the NSDs to be deleted from the GPFS cluster. Specify the names generated when the NSDs were created. Use the **mmisnsd -F** command to display disk names. If there is more than one disk to be deleted, delimit each name with a semicolon (;) and enclose the list of disk names in quotation marks.

-F DiskFile

Specifies a file containing the names of the NSDs, one per line, to be deleted from the GPFS cluster.

-N Node[,Node]

Specifies the nodes to which the disk is attached. If no nodes are listed, the disk is assumed to be directly attached to the local node.

For general information on how to specify node names, see "Specifying nodes as input to GPFS commands" on page 2.

-p NSDId

Specifies the NSD volume ID of an NSD that needs to be cleared from the disk as indicated by the failure of a previous invocation of the **mmdelnsd** command.

Options

NONE

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmdeInsd** command.

You may issue the **mmdeInsd** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To delete **gpfs47nsd** from the GPFS cluster, issue this command:

```
mmdeInsd "gpfs47nsd"
```

The system displays output similar to:

```
mmdeInsd: Processing disk gpfs47nsd
mmdeInsd: 6027-1371 Propagating the changes to all affected
          nodes. This is an asynchronous process.
```

See also

"mmcrnsd Command" on page 137

"mmlnsd Command" on page 227

Location

/usr/lpp/mmfs/bin

mmdelsnapshot Command

Deletes a GPFS snapshot.

Synopsis

mmdelsnapshot *Device Directory*

Description

Use the **mmdelsnapshot** command to delete a GPFS snapshot.

Once the **mmdelsnapshot** command has been issued, the snapshot is marked for deletion and cannot be recovered.

If the node from which the **mmdelsnapshot** command is issued fails, you must reissue the command from another node in the cluster to complete the deletion. Prior to reissuing a subsequent **mmdelsnapshot** command, the file system may be recovered, mounted, and updates may continue to be made and the **mmcrsnapshot** command may be issued. However, the **mmrestorefs** and **mmdelsnapshot** commands may not be issued on other snapshots until the present snapshot is successfully deleted.

If the **mmdelsnapshot** command is issued while a conflicting command is running, the **mmdelsnapshot** command waits for that command to complete. Conflicting operations include:

1. Other snapshot commands on the same snapshot
2. Adding, deleting, replacing disks in the file system
3. Rebalancing, repairing, reducing disk fragmentation in a file system

Any files open in the snapshot will be forcibly closed. The user will receive an **errno** of **ESTALE** on the next file access.

Parameters

Device

The device name of the file system for which the snapshot is to be deleted. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Directory

The name of the snapshot to be deleted

Options

NONE

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmdelsnapshot** command.

You may issue the **mmdelsnapshot** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To delete the snapshot **snap1**, for the file system **fs1**, issue this command:

```
mmdeletesnapshot fs1 snap1
```

The output is similar to this:

```
Deleting snapshot files...
Delete snapshot snap1 complete, err = 0
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots
```

See also

“mmcrsnapshot Command” on page 142

“mmlssnapshot Command” on page 235

“mmrestorefs Command” on page 262

“mmsnapdir Command” on page 277

Location

/usr/lpp/mmfs/bin

mmdf Command

Queries available file space on a GPFS file system.

Synopsis

mmdf *Device* [-d | -F | -m] [-P *PoolName*]

Description

Use the **mmdf** command to display available file space on a GPFS file system. For each disk in the GPFS file system, the **mmdf** command displays this information, by failure group and storage pool:

- The size of the disk.
- The failure group of the disk.
- Whether the disk is used to hold data, metadata, or both.
- Available space in full blocks.
- Available space in fragments.

Displayed values are rounded down to a multiple of 1024 bytes. If the fragment size used by the file system is not a multiple of 1024 bytes, then the displayed values may be lower than the actual values. This can result in the display of a total value that exceeds the sum of the rounded values displayed for individual disks. The individual values are accurate if the fragment size is a multiple of 1024 bytes.

For the file system, the **mmdf** command displays:

- The total number of inodes and the number available.

The **mmdf** command may be run against a mounted or unmounted file system.

Note: The command is I/O intensive and should be run when the system load is light.

Parameters

Device

The device name of the file system to be queried for available file space. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

Options

- d** List only disks that can hold data.
- F** List the number of inodes and how many of them are free.
- m** List only disks that can hold metadata.
- P *PoolName***
Lists only disks that belong to the requested storage pool.

The default is to list all disks.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user:

1. You may issue the **mmdf** command from any node in the GPFS cluster.
2. When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or **mmchcluster** command, you must ensure:
 - a. Proper authorization is granted to all nodes in the GPFS cluster.
 - b. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the **mmdf** command was issued.

Examples

1. To query all disks in the **fs2** file system that can hold data, issue this command:

```
mmdf fs2 -d
```

The system displays information similar to:

```
| disk      disk size  failure holds   holds  free KB in  free KB in
| name      in KB     group metadata data   full blocks fragments
| -----
| Disks in storage pool: system (Maximum disk size allowed is 833 GB)
| -----
| (pool total)      0                                0 (0%)    0 (0%)
| -----
| Disks in storage pool: spl (Maximum disk size allowed is 359 GB)
| gpfs1002nsd  8897968      1 no      yes  8342016 (94%)  928 (0%)
| -----
| (pool total)  8897968                                8342016 (94%)  928 (0%)
| -----
| (data)        8897968                                8342016 (94%)  928 (0%)
| (metadata)    0                                0 (0%)    0 (0%)
| -----
| (total)       8897968                                8342016 (94%)  928 (0%)
```

2. To query all disks in the **fs2** file system that can hold metadata, issue this command:

```
mmdf fs2 -m
```

The system displays information similar to:

```
| disk      disk size  failure holds   holds  free KB in  free KB in
| name      in KB     group metadata data   full blocks fragments
| -----
| Disks in storage pool: system (Maximum disk size allowed is 833 GB)
| gpfs1001nsd  8897968    4001 yes     no   8738816 (98%) 1520 (0%)
| -----
| (pool total)  8897968                                8738816 (98%) 1520 (0%)
| -----
| Disks in storage pool: spl (Maximum disk size allowed is 359 GB)
| -----
| (pool total)    0                                0 (0%)    0 (0%)
| -----
| (data)          0                                0 (0%)    0 (0%)
| (metadata)     8897968                            8738816 (98%) 1520 (0%)
| -----
| (total)        8897968                            8738816 (98%) 1520 (0%)
```

3. To query **fs1** for inode information, issue this command:

```
mmdf fs1 -F
```

The system displays information similar to:

Inode Information

```
-----  
Number of used inodes:      4084  
Number of free inodes:    3002380  
Number of allocated inodes: 3006464  
Maximum number of inodes: 3006464
```

See also

“mmchfs Command” on page 109

“mmcrfs Command” on page 130

“mmdelfs Command” on page 168

“mmlsfs Command” on page 219

Location

/usr/lpp/mmfs/bin

mmeditACL Command

Creates or changes a GPFS access control list.

Synopsis

mmeditACL [-d] [-k {nfs4 | posix | native}] *Filename*

Description

Use the **mmeditACL** command for interactive editing of the ACL of a file or directory. This command uses the default editor, specified in the EDITOR environment variable, to display the current access control information, and allows the file owner to change it. The command verifies the change request with the user before making permanent changes.

| This command cannot be run from a Windows node.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

For information about NFS V4 ACLs, see Chapter 6, “Managing GPFS access control lists and NFS export,” on page 49 and “NFS and GPFS” on page 57.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, **mmeditACL** returns the ACL in a format consistent with the file system setting, specified using the **-k** flag on the **mmcrfs** or **mmchfs** commands.
 - If the setting is **posix**, the ACL is shown as a traditional ACL.
 - If the setting is **nfs4**, the ACL is shown as an NFS V4 ACL.
 - If the setting is **all**, the ACL is returned in its true form.
2. The command **mmeditACL -k nfs4** always produces an NFS V4 ACL.
3. The command **mmeditACL -k posix** always produces a traditional ACL.
4. The command **mmeditACL -k native** always shows the ACL in its true form regardless of the file system setting.

Table 5 describes how **mmeditACL** works.

*Table 5. The **mmeditACL** command for POSIX and NFS V4 ACLs*

Command	ACL	mmcrfs -k	Display	-d (default)
mmeditACL	posix	posix	Access ACL	Default ACL
mmeditACL	posix	nfs4	NFS V4 ACL	Error[1]
mmeditACL	posix	all	Access ACL	Default ACL
mmeditACL	nfs4	posix	Access ACL[2]	Default ACL[2]
mmeditACL	nfs4	nfs4	NFS V4 ACL	Error[1]
mmeditACL	nfs4	all	NFS V4 ACL	Error[1]
mmeditACL -k native	posix	any	Access ACL	Default ACL
mmeditACL -k native	nfs4	any	NFS V4 ACL	Error[1]
mmeditACL -k posix	posix	any	Access ACL	Default ACL
mmeditACL -k posix	nfs4	any	Access ACL[2]	Default ACL[2]
mmeditACL -k nfs4	any	any	NFS V4 ACL	Error[1]

Table 5. The **mmeditACL** command for POSIX and NFS V4 ACLs (continued)

Command	ACL	mmcrfs -k	Display	-d (default)
[1] NFS V4 ACLs include inherited entries. Consequently, there cannot be a separate default ACL.				
[2] Only the mode entries (owner, group, everyone) are translated. The rwX values are derived from the NFS V4 file mode attribute. Since the NFS V4 ACL is more granular in nature, some information is lost in this translation.				

In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual access control entries can be flagged as being inherited (either by files, directories, both, or neither). Consequently, specifying the **-d** flag for an NFS V4 ACL is an error. By its nature, storing an NFS V4 ACL implies changing the inheritable entries (the GPFS default ACL) as well.

Depending on the file system's **-k** setting (**posix**, **nfs4**, or **all**), **mmeditACL** may be restricted. The **mmeditACL** command is not allowed to store an NFS V4 ACL if **-k posix** is in effect, and is not allowed to store a POSIX ACL if **-k nfs4** is in effect. For more information, see the description of the **-k** flag for the **mmchfs**, **mmcrfs**, and **mmlsfs** commands.

Parameters

Filename

The path name of the file or directory for which the ACL is to be edited. If the **-d** option is specified, *Filename* must contain the name of a directory.

Options

-d Specifies that the default ACL of a directory is to be edited.

-k {nfs4 | posix | native}

nfs4 Always produces an NFS V4 ACL.

posix Always produces a traditional ACL.

native Always shows the ACL in its true form regardless of the file system setting.

This option should not be used for routine ACL manipulation. It is intended to provide a way to show the translations that are done. For example, if a **posix** ACL is translated by NFS V4. Beware that if the **-k nfs4** flag is used, but the file system does not allow NFS V4 ACLs, you will not be able to store the ACL that is returned. If the file system does support NFS V4 ACLs, the **-k nfs4** flag is an easy way to convert an existing **posix** ACL to **nfs4** format.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You may issue the **mmeditACL** command only from a node in the GPFS cluster where the file system is mounted.

The **mmeditACL** command may be used to display an ACL. POSIX ACLs may be displayed by any user with access to the file or directory. NFS V4 ACLs have a **READ_ACL** permission that is required for non-privileged users to be able to see an ACL. To change an existing ACL, the user must either be the owner, the root user, or someone with control permission (**WRITE_ACL** is required where the existing ACL is of type NFS V4).

Examples

To edit the ACL for a file named **project2.history**, issue this command:

```
mmeditACL project2.history
```

The current ACL entries are displayed using the default editor, provided that the EDITOR environment variable specifies a complete path name. When the file is saved, the system displays information similar to:

```
mmeditACL: 6027-967 Should the modified ACL be applied? (yes) or (no)
```

After responding **yes**, the ACLs are applied.

See also

“mmdelACL Command” on page 160

“mmgetACL Command” on page 196

“mmputACL Command” on page 246

Location

/usr/lpp/mmfs/bin

mmedquota Command

Sets quota limits.

Synopsis

```
mmedquota {-u [-p ProtoUser] User... | -g [-p ProtoGroup] Group... | -j [-p ProtoFileset] Device:Fileset... |  
-d {-u User... | -g Group... | -j Device:Fileset...} | -t {-u | -g | -j}}
```

Description

The **mmedquota** command serves two purposes:

1. Sets or changes quota limits or grace periods for users, groups, and filesets in the cluster from which the command is issued.
2. Reestablishes user, group, or fileset default quotas for all file systems with default quotas enabled in the cluster.

The **mmedquota** command displays the current values for these limits, if any, and prompts you to enter new values using your default editor:

- Current block usage (display only)
- Current inode usage (display only)
- Inode soft limit
- Inode hard limit
- Block soft limit

Displayed in **KB**, but may be specified using **g**, **G**, **k**, **K**, **m**, or **M**. If no suffix is provided, the number is assumed to be in **bytes**.

- Block hard limit

Displayed in **KB**, but may be specified using **g**, **G**, **k**, **K**, **m**, or **M**. If no suffix is provided, the number is assumed to be in **bytes**.

Note: A block or inode limit of 0 indicates no limit.

The **mmedquota** command waits for the edit window to be closed before checking and applying new values. If an incorrect entry is made, you must reissue the command and enter the correct values.

You can also use the **mmedquota** command to change the file system-specific grace periods for block and file usage if the default of one week is unsatisfactory. The grace period is the time during which users can exceed the soft limit. If the user, group, or fileset does not show reduced usage below the soft limit before the grace period expires, the soft limit becomes the new hard limit.

When setting quota limits for a file system, replication within the file system should be considered. GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported by both the **mmquota** command and the **mmrepquota** command are double the value reported by the **ls** command.

The EDITOR environment variable must contain a complete path name, for example:

```
export EDITOR=/bin/vi
```

- | This command cannot be run from a Windows node.

Parameters

User Name or user ID of target user for quota editing.

Group Name or group ID of target group for quota editing.

Options

- d** Reestablish default quota limits for a specific user, group, or fileset that has had an explicit quota limit set by a previous invocation of the **mmedquota** command.
- g** Sets quota limits or grace times for groups.
- j** Sets quota limits or grace times for filesets.
- p** Applies already-established limits to a particular user, group or fileset.

When invoked with the **-u** option, *ProtoUser* limits are automatically applied to the specified *User* or space-delimited list of users.

When invoked with the **-g** option, *ProtoGroup* limits are automatically applied to the specified *Group* or space-delimited list of groups.

When invoked with the **-j** option, *ProtoFileset* limits are automatically applied to the specified fileset or space-delimited list of fileset names.

You can specify any user as a *ProtoUser* for another *User*, or any group as a *ProtoGroup* for another *Group*, or any fileset as a *ProtoFileset* for another *Fileset*.
- t** Sets grace period during which quotas can exceed the soft limit before it is imposed as a hard limit. The default grace period is one week.

This flag is followed by one of the following flags: **-u**, **-g** or **-j**, to specify whether the changes apply to users, groups, or filesets respectively.
- u** Sets quota limits or grace times for users.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmedquota** command.

GPFS must be running on the node from which the **mmedquota** command is issued.

Examples

1. To set user quotas for userid **paul**, issue this command:

```
mmedquota -u paul
```

The system displays information in your default editor similar to:

```
*** Edit quota limits for USR paul:  
NOTE: block limits will be rounded up to the next multiple of  
the block size. block units may be: K, M, or G  
gpfs0: blocks in use: 864K, limits (soft = 2500K , hard = 10M)  
inodes in use: 9, limits (soft = 100, hard = 1000)
```

2. To reset default group quota values for the group **blueteam**, issue this command:

```
mmedquota -d -g blueteam
```

To verify the change, issue this command:

```
mmrepquota -q fs1
```

The system displays information similar to:

```
fs1: USR quota is on; default quota is on
fs1: GRP quota is on; default quota is on
fs1: FILESET quota is on; default quota is off
```

3. To change the grace periods for all users, issue this command:

```
mmedquota -t -u
```

The system displays information in your default editor similar to:

```
*** Edit grace times:
Time units may be : days, hours, minutes, or seconds
Grace period before enforcing soft limits for USRs:
gpfs0: block grace period: 7 days, file grace period: 7 days
```

See also

“mmcheckquota Command” on page 104

“mmdefedquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmlsquota Command” on page 232

“mmquotaon Command” on page 251

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmexportfs Command

Retrieves the information needed to move a file system to a different cluster.

Synopsis

mmexportfs {*Device* | **all**} **-o** *ExportFilesysData*

Description

The **mmexportfs** command, in conjunction with the **mmimportfs** command, can be used to move one or more GPFS file systems from one GPFS cluster to another GPFS cluster, or to temporarily remove file systems from the cluster and restore them at a later time. The **mmexportfs** command retrieves all relevant file system and disk information and stores it in the file specified with the **-o** parameter. This file must later be provided as input to the **mmimportfs** command. When running the **mmexportfs** command, the file system must be unmounted on all nodes.

When **all** is specified in place of a file system name, any disks that are not associated with a file system will be exported as well.

Exported file systems remain unusable until they are imported back with the **mmimportfs** command to the same or a different GPFS cluster.

Results

Upon successful completion of the **mmexportfs** command, all configuration information pertaining to the exported file system and its disks is removed from the configuration data of the current GPFS cluster and is stored in the user specified file *ExportFilesysData*.

Parameters

Device | **all**

The device name of the file system to be exported. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**. Specify **all** to export all GPFS file systems, as well as all disks that do not belong to a file system yet. This must be the first parameter.

-o *ExportFilesysData*

The path name of a file to which the file system information is to be written. This file must be provided as input to the subsequent **mmimportfs** command.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmexportfs** command.

You may issue the **mmexportfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.

2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To export all file systems in the current cluster, issue this command:

```
mmexportfs all -o /u/admin/exportfile
```

The output is similar to this:

```
mmexportfs: Processing file system fs1 ...
```

```
mmexportfs: Processing file system fs2 ...
```

```
mmexportfs: Processing disks that do not belong to any file system ...
```

```
mmexportfs: 6027-1371 Propagating the changes to all affected  
nodes. This is an asynchronous process.
```

See also

“mmimportfs Command” on page 202

Location

/usr/lpp/mmfs/bin

mmfsck Command

Checks and repairs a GPFS file system.

Synopsis

mmfsck *Device* [-n | -y] [-c | -o] [-t *Directory*] [-v | -V] [-N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}]

The file system must be unmounted before you can run the **mmfsck** command with any option other than **-o**.

Description

The **mmfsck** command in offline mode is intended to be used only in situations where there have been disk or communications failures that have caused **MMFS_FSSTRUCT** error log entries to be issued, or where it is known that disks have been forcibly removed or otherwise permanently unavailable for use in the file system, and other unexpected symptoms are seen by users. In general it is unnecessary to run **mmfsck** in offline mode unless under the direction of the IBM Support Center.

If neither the **-n** nor **-y** flag is specified, the **mmfsck** command runs interactively prompting you for permission to repair each consistency error as reported. It is suggested that in all but the most severely damaged file systems, you run the **mmfsck** command interactively (the default).

The occurrence of I/O errors, or the appearance of a message telling you to run the **mmfsck** command, may indicate file system inconsistencies. If either situation occurs, use the **mmfsck** command to check file system consistency and interactively repair the file system.

For information about file system maintenance and repair, see “Checking and repairing a file system” on page 20. The **mmfsck** command checks for these inconsistencies:

- Blocks marked allocated that do not belong to any file. The corrective action is to mark the block free in the allocation map.
- Files for which an inode is allocated and no directory entry exists (orphaned files). The corrective action is to create directory entries for these files in a **lost+found** subdirectory of the fileset to which the orphaned file or directory belongs. The index number of the inode is assigned as the name. If you do not allow the **mmfsck** command to reattach an orphaned file, it asks for permission to delete the file.
- Directory entries pointing to an inode that is not allocated. The corrective action is to remove the directory entry.
- Incorrectly formed directory entries. A directory file contains the inode number and the generation number of the file to which it refers. When the generation number in the directory does not match the generation number stored in the file’s inode, the corrective action is to remove the directory entry.
- Incorrect link counts on files and directories. The corrective action is to update them with accurate counts.
- Invalid policy files. The corrective action is to delete the file.
- Various problems related to filesets: missing or corrupted fileset metadata, inconsistencies in directory structure related to filesets, missing or corrupted fileset root directory, other problems in internal data structures.

If you are repairing a file system due to node failure and the file system has quotas enabled, it is suggested that you run the **mmcheckquota** command to recreate the quota files.

Indications leading you to the conclusion that you should run the **mmfsck** command include:

- An **MMFS_FSSTRUCT** along with an **MMFS_SYSTEM_UNMOUNT** error log entry on any node indicating some critical piece of the file system is inconsistent.
- Disk media failures

- Partial disk failure
- **EVALIDATE=214**, Invalid checksum or other consistency check failure on a disk data structure, reported in error logs or returned to an application.

For further information on recovery actions and how to contact the IBM Support Center, see the *General Parallel File System: Problem Determination Guide*.

If you are running the online **mmfsck** command to free allocated blocks that do not belong to any files, plan to make file system repairs when system demand is low. This is an I/O intensive activity and it can affect system performance.

Results

If the file system is inconsistent, the **mmfsck** command displays information about the inconsistencies and (depending on the option entered) may prompt you for permission to repair them. The **mmfsck** command tries to avoid actions that may result in loss of data. In some cases, however, it may indicate the destruction of a damaged file.

If there are no file system inconsistencies to detect, the **mmfsck** command reports this information for the file system:

- Number of files
- Used blocks
- Free blocks

All corrective actions, with the exception of recovering lost disk blocks (blocks that are marked as allocated but do not belong to any file), require that the file system be unmounted on all nodes. If the **mmfsck** command is run on a mounted file system, lost blocks are recovered but any other inconsistencies are only reported, not repaired.

If a bad disk is detected, the **mmfsck** command stops the disk and writes an entry to the error log. The operator must manually start and resume the disk when the problem is fixed.

The file system must be unmounted on all nodes before the **mmfsck** command can repair file system inconsistencies.

Parameters

Device

The device name of the file system to be checked and repaired. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

-N {Node[,Node...] | NodeFile | NodeClass}

Specify the nodes to participate in the check and repair of the file system. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the check and repair of the file system).

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

- c** When the file system log has been lost and the file system is replicated, this option specifies that the **mmfsck** command attempt corrective action by comparing the replicas of metadata and data. If this error condition occurs, it is indicated by an error log entry. The **-c** and **-o** flags are mutually exclusive.

- n** Specifies a **no** response to all prompts from the **mmfsck** command. The option reports inconsistencies but it does not change the file system. To save this information, redirect it to an output file when you issue the **mmfsck** command.
- o** Specifies that the file system can be mounted during the operation of the **mmfsck** command. Online mode does not perform a full file system consistency check, but blocks marked as allocated that do not belong to a file are recovered. The **-c** and **-o** flags are mutually exclusive.
- y** Specifies a **yes** response to all prompts from the **mmfsck** command. Use this option only on severely damaged file systems. It allows the **mmfsck** command to take any action necessary for repairs.
- t** *Directory*
Specifies the directory that GPFS uses for temporary storage during **mmfsck** command processing. Although you can issue the command from any node, you must specify a temporary storage directory on the file system manager node. In addition to the location requirement, the storage directory has a minimum space requirement. The minimum space required (in bytes) is equal to the maximum number of inodes in the file system multiplied by 12. The default directory for **mmfsck** processing is **/tmp**.
- v** Specifies that the output is verbose.
- V** Specifies that the output is verbose and contains information for debugging purposes.

Exit status

- 0** Successful completion.
- 2** The command was interrupted before it completed checks or repairs.
- 4** The command changed the file system and it must now be restarted.
- 8** The file system contains damage that has not been repaired.

Security

You must have root authority to run the **mmfsck** command.

You may issue the **mmfsck** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To run the **mmfsck** command on the **fs1** file system, receive a report, but not fix inconsistencies, issue this command:

```
mmfsck fs1 -n
```

The system displays information similar to:

```
Checking "fs1"
Checking inodes
Checking inode map file
Checking directories and files
Checking log files
Checking extended attributes file
```

```
Checking allocation summary file
Checking policy file
Validated policy 'for stripe group fs1': parsed 3 Placement
Rules, 0 Migrate/Delete/Exclude Rules
Checking filesets metadata
Checking file reference counts
Checking file system replication status
```

```
1212416 inodes
  87560 allocated
    0 repairable
    0 repaired
    0 damaged
    0 deallocated
    0 orphaned
    0 attached
```

```
7211746 subblocks
 227650 allocated
    0 unreferenced
    0 deletable
    0 deallocated
```

```
44504 addresses
    0 suspended
```

File system is clean.

mmfsck found no inconsistencies in this file system.

2. To run the **mmfsck** command on the **/dev/fs2** file system, receive a report, and fix inconsistencies, issue this command:

```
mmfsck /dev/fs2 -y
```

The system displays information similar to:

```
Checking "/dev/fs2"
Checking inodes
Checking inode map file
Checking directories and files
Checking log files
Checking extended attributes file
Checking file reference counts
```

```
File inode 6912 is not referenced by any directory.
Reattach inode to lost+found? yes
Checking file system replication status
```

```
33792 inodes
  46  allocated
    0  repairable
    0  repaired
    0  damaged
    0  deallocated
    1  orphaned
    1  attached
```

```
3332520 subblocks
 19762  allocated
    0  unreferenced
    0  deletable
    0  deallocated
```

```
728 addresses
    0  suspended
```

File system is clean.

See also

“mmcheckquota Command” on page 104

“mmcrfs Command” on page 130

“mmdelfs Command” on page 168

“mmdf Command” on page 176

“mmisfs Command” on page 219

Location

/usr/lpp/mmfs/bin

mmfsctl Command

Issues a file system control request.

Synopsis

mmfsctl *Device* {**suspend** | **resume**}

Or,

mmfsctl *Device* {**exclude** | **include**} {-**d** "*DiskName*[:*DiskName*...]" | -**F** *DiskFile* | -**G** *FailureGroup*}

Or,

mmfsctl *Device* **syncFSconfig** {-**n** *RemoteNodesFile* | -**C** *remoteClusterName*} [-**S** *SpecFile*]

Description

Use the **mmfsctl** command to issue control requests to a particular GPFS file system. The command is used to temporarily suspend the processing of all application I/O requests, and later resume them, as well as to synchronize the file system's configuration state between peer clusters in disaster recovery environments.

See *Establishing disaster recovery for your GPFS cluster* in *General Parallel File System: Advanced Administration Guide*.

Before creating a FlashCopy® image of the file system, the user must run **mmfsctl suspend** to temporarily quiesce all file system activity and flush the internal buffers on all nodes that mount this file system. The on-disk metadata will be brought to a consistent state, which provides for the integrity of the FlashCopy snapshot. If a request to the file system is issued by the application after the invocation of this command, GPFS suspends this request indefinitely, or until the user issues **mmfsctl resume**.

Once the FlashCopy image has been taken, the **mmfsctl resume** command can be issued to resume the normal operation and complete any pending I/O requests.

The **mmfsctl syncFSconfig** command extracts the file system's related information from the local GPFS configuration data, transfers this data to one of the nodes in the peer cluster, and attempts to import it there.

Once the GPFS file system has been defined in the primary cluster, users run this command to import the configuration of this file system into the peer recovery cluster. After producing a FlashCopy image of the file system and propagating it to the peer cluster using Peer-to-Peer Remote Copy (PPRC), users similarly run this command to propagate any relevant configuration changes made in the cluster after the previous snapshot.

The primary cluster configuration server of the peer cluster must be available and accessible using remote shell and remote copy at the time of the invocation of the **mmfsctl syncFSconfig** command. Also, the peer GPFS clusters should be defined to use the same remote shell and remote copy mechanism, and they must be set up to allow nodes in peer clusters to communicate without the use of a password.

Not all administrative actions performed on the file system necessitate this type of resynchronization. It is required only for those actions that modify the file system information maintained in the local GPFS configuration data, which includes:

- Additions, removals, and replacements of disks (commands **mmadddisk**, **mmdeldisk**, **mmrpldisk**)
- Modifications to disk attributes (command **mmchdisk**)

- Changes to the file system's mount point (command **mmchfs -T**)
- Changing the file system device name (command **mmchfs -W**)

The process of synchronizing the file system configuration data can be automated by utilizing the **syncfsconfig** user exit.

The **mmfsctl exclude** command is to be used only in a disaster recovery environment, only after a disaster has occurred, and only after ensuring that the disks in question have been physically disconnected. Otherwise, unexpected results may occur.

The **mmfsctl exclude** command can be used to manually override the file system descriptor quorum after a site-wide disaster. See *Establishing disaster recovery for your GPFS cluster* in *General Parallel File System: Advanced Administration Guide*. This command enables users to restore normal access to the file system with less than a quorum of available file system descriptor replica disks, by effectively excluding the specified disks from all subsequent operations on the file system descriptor. After repairing the disks, the **mmfsctl include** command can be issued to restore the initial quorum configuration.

Parameters

Device

The device name of the file system. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**. If **all** is specified with the **syncFSconfig** option, this command is performed on all GPFS file systems defined in the cluster.

exclude

Instructs GPFS to exclude the specified group of disks from all subsequent operations on the file system descriptor, and change their availability state to **down**, if the conditions in the Note below are met.

If necessary, this command assigns additional disks to serve as the disk descriptor replica holders, and migrate the disk descriptor to the new replica set. The excluded disks are not deleted from the file system, and still appear in the output of the **mmlsdisk** command.

Note: The **mmfsctl exclude** command is to be used only in a disaster recovery environment, only after a disaster has occurred, and only after ensuring that the disks in question have been physically disconnected. Otherwise, unexpected results may occur.

include

Informs GPFS that the previously excluded disks have become operational again. This command writes the up-to-date version of the disk descriptor to each of the specified disks, and clears the **excl** tag.

resume

Instructs GPFS to resume the normal processing of I/O requests on all nodes.

suspend

Instructs GPFS to flush the internal buffers on all nodes, bring the file system to a consistent state on disk, and suspend the processing of all subsequent application I/O requests.

syncFSconfig

Synchronizes the configuration state of a GPFS file system between the local cluster and its peer in two-cluster disaster recovery configurations.

-C remoteClusterName

Specifies the name of the GPFS cluster that owns the remote GPFS file system.

-d "DiskName[:DiskName...]"

Specifies the names of the NSDs to be included or excluded by the **mmfsctl** command. Separate the names with semicolons (;) and enclose the list of disk names in quotation marks.

-F *DiskFile*

Specifies a file containing the names of the NSDs, one per line, to be included or excluded by the **mmfsctl** command.

-G *FailureGroup*

A number identifying the failure group for disks to be included or excluded by the **mmfsctl** command.

-n *RemoteNodesFile*

Specifies a list of contact nodes in the peer recovery cluster that GPFS uses when importing the configuration data into that cluster. Although any node in the peer cluster can be specified here, users are advised to specify the identities of the peer cluster's primary and secondary cluster configuration servers, for efficiency reasons.

-S *SpecFile*

Specifies the description of changes to be made to the file system, in the peer cluster during the import step. The format of this file is identical to that of the *ChangeSpecFile* used as input to the **mmimportfs** command. This option can be used, for example, to define the assignment of the NSD servers for use in the peer cluster.

Options

None.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Results

The **mmfsctl** command returns 0 if successful.

Security

You must have root authority to run the **mmfsctl** command.

You may issue the **mmfsctl** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This sequence of commands creates a FlashCopy image of the file system and propagates this image to the recovery cluster using the Peer-to-Peer Remote Copy technology. The following configuration is assumed:

Site	LUNs
Primary cluster (site A)	lunA1, lunA2
Recovery cluster (site B)	lunB1

lunA1 FlashCopy source

lunA2 FlashCopy target, PPRC source

lunB1 PPRC target

A single GPFS file system named **fs0** has been defined in the primary cluster over lunA1.

1. In the primary cluster, suspend all file system I/O activity and flush the GPFS buffers

```
mmfsctl fs0 suspend
```

The output is similar to this:

```
Writing dirty data to disk
Quiescing all file system operations
Writing dirty data to disk again
```

2. Establish a FlashCopy pair using lunA1 as the source and lunA2 as the target.
3. Resume the file system I/O activity:

```
mmfsctl fs0 resume
```

The output is similar to this:

```
Resuming operations.
```

4. Establish a Peer-to-Peer Remote Copy (PPRC) path and a synchronous PPRC volume pair lunA2-lunB1 (primary-secondary). Use the 'copy entire volume' option and leave the 'permit read from secondary' option disabled.
5. Wait for the completion of the FlashCopy background task. Wait for the PPRC pair to reach the duplex (fully synchronized) state.
6. Terminate the PPRC volume pair lunA2-lunB1.
7. If this is the first time the snapshot is taken, or if the configuration state of **fs0** changed since the previous FlashCopy snapshot, propagate the most recent configuration to site B:

```
mmfsctl fs0 syncFSconfig -n recovery_clust_nodelist
```

Location

/usr/lpp/mmfs/bin

mmgetacl Command

Displays the GPFS access control list of a file or directory.

Synopsis

mmgetacl [-d] [-o *OutFilename*] [-k {nfs4 | posix | native}] *Filename*

Description

Use the **mmgetacl** command to display the ACL of a file or directory.

For information about NFS V4 ACLs, see Chapter 6, “Managing GPFS access control lists and NFS export,” on page 49 and “NFS and GPFS” on page 57.

Users may need to see ACLs in their true form as well as how they are translated for access evaluations. There are four cases:

1. By default, **mmgetacl** returns the ACL in a format consistent with the file system setting, specified using the **-k** flag on the **mmcrfs** or **mmchfs** commands.
If the setting is **posix**, the ACL is shown as a traditional ACL.
If the setting is **nfs4**, the ACL is shown as an NFS V4 ACL.
If the setting is **all**, the ACL is returned in its true form.
2. The command **mmgetacl -k nfs4** always produces an NFS V4 ACL.
3. The command **mmgetacl -k posix** always produces a traditional ACL.
4. The command **mmgetacl -k native** always shows the ACL in its true form regardless of the file system setting.

| This command cannot be run from a Windows node.

Table 6 describes how **mmgetacl** works.

*Table 6. The **mmgetacl** command for POSIX and NFS V4 ACLs*

Command	ACL	mmcrfs -k	Display	-d (default)
mmgetacl	posix	posix	Access ACL	Default ACL
mmgetacl	posix	nfs4	NFS V4 ACL	Error[1]
mmgetacl	posix	all	Access ACL	Default ACL
mmgetacl	nfs4	posix	Access ACL[2]	Default ACL[2]
mmgetacl	nfs4	nfs4	NFS V4 ACL	Error[1]
mmgetacl	nfs4	all	NFS V4 ACL	Error[1]
mmgetacl -k native	posix	any	Access ACL	Default ACL
mmgetacl -k native	nfs4	any	NFS V4 ACL	Error[1]
mmgetacl -k posix	posix	any	Access ACL	Default ACL
mmgetacl -k posix	nfs4	any	Access ACL[2]	Default ACL[2]
mmgetacl -k nfs4	any	any	NFS V4 ACL	Error[1]

[1] NFS V4 ACLs include inherited entries. Consequently, there cannot be a separate default ACL.

[2] Only the mode entries (owner, group, everyone) are translated. The **rwX** values are derived from the NFS V4 file mode attribute. Since the NFS V4 ACL is more granular in nature, some information is lost in this translation.

Parameters

Filename

The path name of the file or directory for which the ACL is to be displayed. If the **-d** option is specified, *Filename* must contain the name of a directory.

Options

-d Specifies that the default ACL of a directory is to be displayed.

-k {nfs4 | posix | native}

nfs4 Always produces an NFS V4 ACL.

posix Always produces a traditional ACL.

native Always shows the ACL in its true form regardless of the file system setting.

-o OutFilename

The path name of a file to which the ACL is to be written.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have read access to the directory where the file exists to run the **mmgetacl** command.

You may issue the **mmgetacl** command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To display the ACL for a file named **project2.history**, issue this command:

```
mmgetacl project2.history
```

The system displays information similar to:

```
#owner:paul
#group:design
user::rwx
group::r-x-
other::r-x-
```

2. This is an example of an NFS V4 ACL displayed using **mmgetacl**. Each entry consists of three lines reflecting the greater number of permissions in a text format. An entry is either an **allow** entry or a **deny** entry. An **X** indicates that the particular permission is selected, a minus sign (-) indicates that is it not selected. The following access control entry explicitly allows **READ**, **EXECUTE** and **READ_ATTR** to the **staff** group on a file:

```
group:staff:r-x-:allow
(X)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (X)READ_ATTR (-)READ_NAMED
(-)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

3. This is an example of a directory ACLs, which may include *inherit* entries (the equivalent of a default ACL). These do not apply to the directory itself, but instead become the initial ACL for any objects created within the directory. The following access control entry explicitly denies **READ/LIST**, **READ_ATTR**, and **EXEC/SEARCH** to the **sys** group.

```
group:sys:----:deny:DirInherit
(X)READ/LIST (-)WRITE/CREATE (-)MKDIR (-)SYNCHRONIZE (-)READ_ACL (X)READ_ATTR (-)READ_NAMED
(-)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

See also

“mmeditacI Command” on page 179

“mmdeIacI Command” on page 160

“mmputacI Command” on page 246

Location

/usr/lpp/mmfs/bin

mmgetstate Command

Displays the state of the GPFS daemon on one or more nodes.

Synopsis

mmgetstate [-L] [-s] [-v] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]

Description

Use the **mmgetstate** command to show the state of the GPFS daemon on one or more nodes.

Parameters

- | **-a** Show the state of the GPFS daemon on all nodes in the cluster.
- N {Node[,Node...] | NodeFile | NodeClass}**
Directs the **mmgetstate** command to return GPFS daemon information for a set of nodes. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

Options

- L** Display quorum, number of nodes up, total number of nodes, and other extended node information.
- s** Display summary information such as: number of local and remote nodes that have joined in the cluster, number of quorum nodes.
- v** Display intermediate error messages.

The GPFS states recognized and displayed by this command are:

active GPFS is ready for operations.

arbitrating

A node is trying to form a quorum with the other available nodes.

down GPFS daemon is not running on the node.

unknown

Unknown value. Node cannot be reached or some other error occurred.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmgetstate** command.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.

2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To display the quorum, the number of nodes up, and the total number of nodes for the GPFS cluster, issue:

```
mmgetstate -a -L
```

The system displays output similar to:

Node number	Node name	Quorum	Nodes up	Total nodes	GPFS state	Remarks
1	c5n92	3	5	12	active	
2	c5n94	3	5	12	active	
3	c5n95	3	5	12	active	quorum node
4	c5n96	3	5	12	active	
5	c5n97	3	5	12	active	quorum node
6	c5n98	3	5	12	active	
7	c5n107	3	5	12	active	quorum node
8	c5n108	3	5	12	active	
9	c5n109	3	5	12	active	quorum node
10	c5n110	3	5	12	down	
11	c5n111	3	5	12	active	quorum node
12	c5n112	3	5	12	active	

The 3 under the Quorum column means that you must have three quorum nodes up to achieve quorum.

2. This is an example of a cluster using node quorum with tiebreaker disks. Note the * in the Quorum field, which indicates that tiebreaker disks are being used:

```
mmgetstate -a -L
```

The system displays output similar to:

Node number	Node name	Quorum	Nodes up	Total nodes	GPFS state	Remarks
1	k5n91	5*	8	21	active	
2	k5n92	5*	8	21	active	quorum node
3	k5n94	5*	8	21	active	
5	k5n96	5*	8	21	active	
6	k5n97	5*	8	21	active	quorum node
7	k5n98	5*	8	21	active	
8	k5n99	5*	8	21	active	quorum node

3. To display summary information, issue this command:

```
mmgetstate -s
```

The system displays output similar to:

Node number	Node name	GPFS state
5	c5n97	active

Summary information

```
-----
Number of nodes defined in the cluster: 12
Number of local nodes active in the cluster: 12
Number of remote nodes joined in this cluster: 0
Number of quorum nodes defined in the cluster: 5
Number of quorum nodes active in the cluster: 5
Quorum = 3, Quorum achieved
```

See also

“mmchconfig Command” on page 92

“mmcrcluster Command” on page 124

“mmshutdown Command” on page 275

“mmstartup Command” on page 280

Location

/usr/lpp/mmfs/bin

mmimportfs Command

Imports into the cluster one or more file systems that were created in another GPFS cluster.

Synopsis

mmimportfs {*Device* | **all**} **-i** *ImportfsFile* [**-S** *ChangeSpecFile*]

Description

The **mmimportfs** command, in conjunction with the **mmexportfs** command, can be used to move into the current GPFS cluster one or more file systems that were created in another GPFS cluster. The **mmimportfs** command extracts all relevant file system and disk information from the *ExportFilesysData* file specified with the **-i** parameter. This file must have been created by the **mmexportfs** command.

When **all** is specified in place of a file system name, any disks that are not associated with a file system will be imported as well.

If the file systems being imported were created on nodes that do not belong to the current GPFS cluster, the **mmimportfs** command assumes that all disks have been properly moved, and are online and available to the appropriate nodes in the current cluster.

If any node in the cluster, including the node on which you are running the **mmimportfs** command, does not have access to one or more disks, use the **-S** option to assign NSD servers to those disks.

The **mmimportfs** command attempts to preserve any NSD server assignments that were in effect when the file system was exported.

If the file system was exported from a cluster created with a version of GPFS prior to 2.3, it is possible that the disks of the file system are not NSDs. Such disks will be automatically converted into NSDs by the **mmimportfs** command.

After the **mmimportfs** command completes, use **mmlnsd** to display the NSD server names that are assigned to each of the disks in the imported file system. Use **mmchnsd** to change the current NSD server assignments as needed.

After the **mmimportfs** command completes, use **mmlsdisk** to display the failure groups to which each disk belongs. Use **mmchdisk** to make adjustments if necessary.

If you are importing file systems into a cluster that already contains GPFS file systems, it is possible to encounter name conflicts. You must resolve such conflicts before the **mmimportfs** command can succeed. You can use the **mmchfs** command to change the device name and mount point of an existing file system. If there are disk name conflicts, use the **mmcrnsd** command to define new disks and specify unique names (rather than let the command generate names). Then replace the conflicting disks using **mmrpldisk** and remove them from the cluster using **mmdelnsd**.

Results

Upon successful completion of the **mmimportfs** command, all configuration information pertaining to the file systems being imported is added to configuration data of the current GPFS cluster.

Parameters

Device | **all**

The device name of the file system to be imported. File system names need not be fully-qualified.

fs0 is as acceptable as **/dev/fs0**. Specify **all** to import all GPFS file systems, as well as all disks that do not belong to a file system yet. This must be the first parameter.

-i ImportfsFile

The path name of the file containing the file system information. This file must have previously been created with the **mmexportfs** command.

-S ChangeSpecFile

The path name of an optional file containing disk descriptors, one per line, in the format:

DiskName:ServerList

DiskName

Is the name of a disk from the file system being imported.

ServerList

Is a comma separated list of NSD server nodes. You can specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list. If you do not define a server list, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a server list.

When specifying server nodes for your NSDs, the output of the **mmiscluster** command lists the host name and IP address combinations recognized by GPFS. The utilization of aliased host names not listed in the **mmiscluster** command output may produce undesired results.

If you do not define a *ServerList*, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a *ServerList*.

Note:

1. You cannot change the name of a disk. You cannot change the disk usage or failure group assignment with the **mmimportfs** command. Use the **mmchdisk** command for this purpose.
2. All disks that do not have descriptors in *ChangeSpecFile* are assigned the NSD servers that they had at the time the file system was exported. All disks with NSD servers that are not valid are assumed to be SAN-attached to all nodes in the cluster. Use the **mmchnsd** command to assign new or change existing NSD server nodes.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmimportfs** command.

You may issue the **mmimportfs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To import all file systems in the current cluster, issue this command:

```
mmimportfs all -i /u/admin/exportfile
```

The output is similar to this:

```
mmimportfs: Processing file system fs1 ...
mmimportfs: Processing disk gpfs2nsd
mmimportfs: Processing disk gpfs3nsd
mmimportfs: Processing disk gpfs4nsd

mmimportfs: Processing file system fs2 ...
mmimportfs: Processing disk gpfs1nsd1
mmimportfs: Processing disk gpfs5nsd

mmimportfs: Processing disks that do not belong to any file system ...
mmimportfs: Processing disk gpfs6nsd
mmimportfs: Processing disk gpfs1001nsd

mmimportfs: Committing the changes ...

mmimportfs: The following file systems were successfully imported:
    fs1
    fs2
mmimportfs: 6027-1371 Propagating the changes to all affected
                nodes. This is an asynchronous process.
```

See also

“mmexportfs Command” on page 185

Location

/usr/lpp/mmfs/bin

mmlinkfileset Command

Creates a junction that references the root directory of a GPFS fileset.

Synopsis

mmlinkfileset *Device FilesetName* [-J *JunctionPath*]

Description

The **mmlinkfileset** command creates a junction at *JunctionPath* that references the root directory of *FilesetName*. The junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset, the parent, to the root directory of a child fileset. From the user's viewpoint, a junction always appears as if it were a directory, but the user is not allowed to issue the **unlink** or **rmdir** commands on a junction. Instead, the **mmunlinkfileset** command must be used to remove a junction.

If *JunctionPath* is not specified, the junction is created in the current directory with the name *FilesetName*. The user may use the **mv** command on the directory to move to a new location in the parent fileset, but the **mv** command is not allowed to move the junction to a different fileset.

For information on GPFS filesets, see the chapter *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system that contains the fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

FilesetName

Specifies the name of the fileset to be linked. It must not already be linked into the namespace.

-J *JunctionPath*

Specifies the name of the junction. The name must not refer to an existing file system object.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmlinkfileset** command.

You may issue the **mmlinkfileset** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This command links fileset **fset1** in file system **gpfs1** to junction path **/gpfs1/fset1**:

```
mmmlinkfileset gpfs1 fset1 -J /gpfs1/fset1
```

The system displays output similar to:

```
Fileset 'fset1' linked at '/gpfs1/fset1'.
```

To confirm the change, issue this command:

```
mmllsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':  
Name           Status    Path  
root           Linked    /gpfs1  
fset1          Linked    /gpfs1/fset1
```

See also

“mmchfileset Command” on page 107

“mmcrfileset Command” on page 128

“mmdelfileset Command” on page 165

“mmllsfileset Command” on page 216

“mmunlinkfileset Command” on page 288

Location

/usr/lpp/mmfs/bin

mmlsattr Command

Queries file attributes.

Synopsis

mmlsattr [-L] *FileName* [*FileName...*]

Description

Use the **mmlsattr** command to display attributes of a file.

Results

For the specified file, the **mmlsattr** command lists:

- Current number of copies of data for a file and the maximum value
- Number of copies of the metadata for a file and the maximum value
- Whether the Direct I/O caching policy is in effect for a file

Parameters

FileName

The name of the file to be queried. You must enter at least one file name.

Options

- L** Displays additional file attributes:
- The file's assigned storage pool name.
 - The name of the fileset that includes the file.
 - The name of the snapshot that includes the file.
If the file is not part of a snapshot, an empty string is displayed.
 - Whether the file is illplaced. This word would appear under **flags**.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have read access to run the **mmlsattr** command.

You may issue the **mmlsattr** command only from a node in the GPFS cluster where the file system is mounted.

Examples

1. To list the attributes of a file named **project4.sched**, issue this command:

```
mmlsattr -L /fs0/project4.sched
```

The system displays information similar to:

```
file name:           /fs0/project4.sched
metadata replication: 1 max 2
data replication:    1 max 2
```

```
flags:
storage pool name:    system
fileset name:        root
snapshot name:
```

2. To show the attributes for all files in the root directory of file system **fs0**, issue this command:

```
mmfssattr /fs0/*
```

The system displays information similar to:

```
      replication factors
metadata(max) data(max) file      [flags]
-----
      1 ( 1)   1 ( 1) /fs0/project4.sched
      1 ( 1)   1 ( 1) /fs0/project4.hist
      1 ( 1)   1 ( 1) /fs0/project5.plan
```

See also

“mmchattr Command” on page 86

Location

/usr/lpp/mmfs/bin

mmlscluster Command

Displays the current configuration information for a GPFS cluster.

Synopsis

mmlscluster [--cnfs]

Description

Use the **mmlscluster** command to display the current configuration information for a GPFS cluster.

For the GPFS cluster, the **mmlscluster** command displays:

- The cluster name
- The cluster ID
- GPFS UID domain
- The remote shell command being used
- The remote file copy command being used
- The primary GPFS cluster configuration server
- The secondary GPFS cluster configuration server
- A list of nodes belonging the GPFS cluster

For each node, the command displays:

- The node number assigned to the node by GPFS
- GPFS daemon node interface name
- Primary network IP address
- GPFS administration node interface name
- Remarks, such as whether the node is a quorum node or not

Parameters

NONE

Options

--cnfs Displays information about clustered NFS.

Exit status

0 Successful completion.

nonzero A failure has occurred.

Security

You must have root authority to run the **mmlscluster** command.

You may issue the **mmlscluster** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To display the current configuration information for the GPFS cluster, issue this command:

```
mmfsccluster
```

The system displays information similar to:

GPFS cluster information

=====

```
GPFS cluster name:      cluster1.kgn.ibm.com
GPFS cluster id:        680681562214606028
GPFS UID domain:        cluster1.kgn.ibm.com
Remote shell command:   /usr/bin/rsh
Remote file copy command: /usr/bin/rcp
```

GPFS cluster configuration servers:

```
Primary server:   k164sn06.kgn.ibm.com
Secondary server: k164n04.kgn.ibm.com
```

Node	Daemon node name	IP address	Admin node name	Designation
1	k164n04.kgn.ibm.com	89.116.68.68	k164n04.kgn.ibm.com	quorum
2	k164n05.kgn.ibm.com	89.116.68.69	k164n05.kgn.ibm.com	quorum
3	k164n06.kgn.ibm.com	89.116.68.70	k164sn06.kgn.ibm.com	quorum-manager

See also

“mmaddnode Command” on page 70

“mmchcluster Command” on page 89

“mmcrcluster Command” on page 124

“mmdelnode Command” on page 170

Location

/usr/lpp/mmfs/bin

mmlsconfig Command

Displays the current configuration data for a GPFS cluster.

Synopsis

mmlsconfig

Description

Use the **mmlsconfig** command to display the current configuration data for a GPFS cluster.

Depending on your configuration, additional information that is set by GPFS may be displayed to assist in problem determination when contacting the IBM Support Center. If a configuration parameter is not shown in the output of this command, the default value for that parameter, as documented in the **mmchconfig** command, is in effect.

Parameters

NONE

Options

NONE

Exit status

0 Successful completion.

nonzero A failure has occurred.

Security

You must have root authority to run the **mmlsconfig** command.

You may issue the **mmlsconfig** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To display the current configuration data for the GPFS cluster that you are running on, issue this command:

```
mmlsconfig
```

The system displays information similar to:

```
Configuration data for cluster cluster1.kgn.ibm.com:
-----
clusterName cluster1.kgn.ibm.com
clusterId 680681562214606028
```

```
clusterType 1c
autoload yes
minReleaseLevel 3.2.0.0
pagepool 512m
maxblocksize 4M
cipherList AUTHONLY
```

File systems in cluster cluster1.kgn.ibm.com:

```
-----
/dev/fs1
/dev/fs2
```

See also

“mmchcluster Command” on page 89

“mmchconfig Command” on page 92

“mmcrcluster Command” on page 124

Location

/usr/lpp/mmfs/bin

mmlsdisk Command

Displays the current configuration and state of the disks in a file system.

Synopsis

mmlsdisk *Device* [-d "*DiskName*;*DiskName*..."] [-e] [-L]

Or,

mmlsdisk *Device* [-d "*DiskName*;*DiskName*..."] {-m | -M}

Description

Use the **mmlsdisk** command to display the current state of the disks in the file system.

The **mmlsdisk** command may be run against a mounted or unmounted file system.

For each disk in the list, the **mmlsdisk** command displays:

- Disk name
- Driver type
- Sector size
- Failure group
- Whether it holds metadata
- Whether it holds data
- Current status:

ready Normal status.

suspended

Indicates that data is to be migrated off this disk.

being emptied

Transitional status in effect while a disk deletion is pending.

replacing

Transitional status in effect for old disk while replacement is pending.

replacement

Transitional status in effect for new disk while replacement is pending.

- Availability:

up Disk is available to GPFS for normal **read** and **write** operations.

down No **read** and **write** operations can be performed on this disk.

recovering

An intermediate state for disks coming up, during which GPFS verifies and corrects data. **read** operations can be performed while a disk is in this state but **write** operations cannot.

unrecovered

The disks was not successfully brought up.

- Disk ID
- Storage pool that the disk is assigned to.

Parameters

Device

The device name of the file system to which the disks belong. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

-d "*DiskName*[:*DiskName*...]"

The name of the disks for which you want to display current configuration and state information. When you enter multiple values for *DiskName*, you must separate them with semicolons and enclose the list in quotation marks.

"gpfs3nsd;gpfs4nsd;gpfs5nsd"

Options

- e** Display all of the disks in the file system that do not have an availability of **up** and a status of **ready**. If all disks in the file system are **up** and **ready**, the message displayed is:
6027-623 All disks up and ready
- L** Displays an extended list of the disk parameters, including the disk ID field and the **remarks** field. The **remarks** column shows the current file system descriptor quorum assignments, and displays the excluded disks. The **remarks** field contains **desc** for all disks assigned as the file system descriptor holders and **excl** for all excluded disks.
- M** Displays whether I/O requests to the disk are satisfied on the local node, or using an NSD server. If the I/O is done using an NSD server, shows the NSD server name and the underlying disk name on that server node.
- m** Displays whether I/O requests to the disk are satisfied on the local node, or using an NSD server. This scope of this options is the node on which the **mmlsdisk** command is issued.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user:

1. You may issue the **mmlsdisk** command from any node in the GPFS cluster.
2. When using the **rsh** and **rlogin** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:
 - a. Proper authorization is granted to all nodes in the GPFS cluster.
 - b. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.
3. As root, the command can also do an **mmlsdisk** on remote file systems.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the **mmlsdisk** command was issued.

The **mmlsdisk** command does not work if GPFS is down.

Examples

1. To display the current state of **gpfs2nsd**, issue this command:

```
mmllsdisk /dev/fs0 -d gpfs2nsd
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	storage pool
gpfs2nsd	nsd	512	4002	yes	yes	ready	up	system

2. To display the current states of **gpfs2nsd**, **gpfs3nsd**, and **gpfs4nsd**, and display their respective disk ids and the descriptor quorum assignment, issue this command:

```
mmllsdisk /dev/fs0 -d "gpfs2nsd;gpfs3nsd;gpfs4nsd" -L
```

The system displays information similar to:

disk name	driver type	sector size	failure group	holds metadata	holds data	status	availability	disk id	storage pool	remarks
gpfs2nsd	nsd	512	4002	yes	yes	ready	up	2	system	desc
gpfs3nsd	nsd	512	4002	yes	yes	ready	up	3	system	
gpfs4nsd	nsd	512	4002	yes	yes	ready	up	4	system	

Number of quorum disks: 3
Read quorum value: 2
Write quorum value: 2

3. To display the whether the I/O is performed locally or using an NSD server, the NSD server name, and the underlying disk name for the file system named **test**, issue this command:

```
mmllsdisk test -M
```

The system displays information similar to:

Disk name	I/O performed on node	Device	Availability
gpfs7nsd	localhost	/dev/gpfs2vsd	up
gpfs10nsd	k5n88.kgn.ibm.com	/dev/gpfs3vsd	up
gpfs4nsd	localhost	/dev/hdisk10	up

4. To display the same information as in the previous example, but limited to the node on which the command is issued, issue this command:

```
mmllsdisk test -m
```

The system displays information similar to:

Disk name	I/O performed on node	Device	Availability
gpfs7nsd	localhost	/dev/gpfs2vsd	up
gpfs10nsd	k5n88.kgn.ibm.com	-	up
gpfs4nsd	localhost	/dev/hdisk10	up

See also

“mmadddisk Command” on page 66

“mmchdisk Command” on page 100

“mmdeldisk Command” on page 162

“mmrpldisk Command” on page 271

Location

/usr/lpp/mmfs/bin

mmlsfileset Command

Displays attributes and status for GPFS filesets.

Synopsis

mmlsfileset *Device* {*[[Fileset [,Fileset ...]] [-J Junction [,Junction ...]] | -F FileName*} **[-L] [-d] [-i]**

Description

Use the **mmlsfileset** command to display information for the filesets that belong to a given GPFS file system. The default is to display information for all filesets in the file system. You may choose to display information for only a subset of the filesets.

The operation of the **-L** flag omits the attributes listed without it, namely status and junction path. In addition, if the fileset has status 'deleted', then **-L** also displays the name of the latest snapshot that includes the fileset in place of the root inode number and parent fileset identifier.

The attributes displayed are:

- Name of the fileset
- Status of the fileset (when the **-L** flag is omitted)
- Junction path to the fileset (when the **-L** flag is omitted)
- Fileset identifier (when the **-L** flag is included)
- Root inode number, if not deleted (when the **-L** flag is included)
- Parent fileset identifier, if not deleted (when the **-L** flag is included)
- Latest including snapshot, if deleted (when the **-L** flag is included)
- Creation time (when the **-L** flag is included)
- Number of inodes in use (when the **-i** flag is included)
- Data size (in KB) (when the **-d** flag is included)
- Comment (when the **-L** flag is included)

For information on GPFS filesets, see the topic *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system that contains the fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

Fileset Specifies a comma-separated list of fileset names.

-J Junction

Specifies a comma-separated list of path names. They are not restricted to fileset junctions, but may name any file or directory within the filesets to be listed.

-F FileName

Specifies the name of a file containing either fileset names or path names. Each line must contain a single entry. All path names must be fully-qualified.

Options

-d Display the number of blocks in use for the fileset.

- i Display the number of inodes in use for the fileset.
- L Display additional information for the fileset. This includes:
 - Fileset identifier
 - Root inode number
 - Parent identifier
 - Fileset creation time
 - User defined comments, if any

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You may issue the **mmfsfileset** command from any node in the GPFS cluster.

When using the **rccp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. This command displays fileset information for all filesets in file system **gpfs1**:

```
mmfsfileset gpfs1
```

The system displays information similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
root      Linked   /gpfs1
fset1     Linked   /gpfs1/fset1
fset2     Linked   /gpfs1/fset1/fset2
```

2. The file **Filesetlist** has this information:

```
fset1
fset2
```

This command displays all filesets in file system **gpfs1** that are listed in **Filesetlist**:

```
mmfsfileset gpfs1 -F Filesetlist
```

The system displays information similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
fset1     Linked   /gpfs1/fset1
fset2     Linked   /gpfs1/fset1/fset2
```

3. These commands displays information for a file system with filesets and snapshots. Note that deleted filesets that are saved in snapshots are displayed with the name enclosed in parentheses.

a. `mmfsfileset fs1 -d -i`

The system displays information similar to:

```

Filesets in file system 'fs1':
Name      Status   Path                               Inodes  Data (in  KB)
root      Linked   /gpfs                               3        53528
(gone)    Deleted  /gpfs/.snapshots/Snap17/gone      0         0
TestF4    Linked   /gpfs/test-f4                      3         24
TestF3    Linked   /gpfs/test-f4/dir1/f3              2         16
TestF2    Unlinked --                               98        784
TestF5    Linked   <TestF2>/subdir/f5                 1          8

```

b. `mmfsfileset fs1 -L`

The system displays information similar to:

```

Filesets in file system 'fs1':
Name  Id  RootInode  ParentId  Created          Comment
root   0      3          --  Thu Feb 16 12:14:38 2006 root fileset
(gone) 1 latest:      Snap17  Thu Feb 16 12:22:37 2006 But not forgotten
TestF4 2    35849       0  Thu Feb 16 12:22:38 2006 Fourth in a series
TestF3 3    75921       2  Thu Feb 16 12:22:39 2006
TestF2 4    75923      --  Thu Feb 16 12:22:39 2006
TestF5 5    76021       4  Thu Feb 16 12:22:39 2006 Number 5

```

c. `mmfsfileset fs1 TestF2,TestF5 -J /gpfs/test-f4/dir1,/gpfs/test-f4/dir1/f3/dir2/...`

The system displays information similar to:

```

Filesets in file system 'fs1':
Name      Status   Path
TestF2    Unlinked --
TestF5    Linked   <TestF2>/subdir/f5
TestF4    Linked   /gpfs/test-f4
TestF3    Linked   /gpfs/test-f4/dir1/f3

```

See also

“`mmchfileset` Command” on page 107

“`mmcrfileset` Command” on page 128

“`mmdefileset` Command” on page 165

“`mmmlinkfileset` Command” on page 205

“`mmunlinkfileset` Command” on page 288

Location

`/usr/lpp/mmfs/bin`

mmlsfs Command

Displays file system attributes.

Synopsis

```
mmlsfs {Device | all | all_local | all_remote} [-A] [-a] [-B] [-D] [-d] [-E] [-F] [-f] [-I] [-i] [-j] [-k] [-K][-L] [-M]
| [-m] [-n] [-o] [-P] [-Q] [-R] [-r] [-S] [-t] [-T] [-u] [-V] [-z]
```

Description

Use the **mmlsfs** command to list the attributes of a file system.

Depending on your configuration, additional information that is set by GPFS may be displayed to assist in problem determination when contacting the IBM Support Center.

Results

If you do not specify any options, all attributes of the file system are displayed. When you specify options, only those attributes specified are listed, in the order issued in the command. Some parameters are preset for optimum performance and, although they display in the **mmlsfs** command output, you cannot change them.

Parameters

Device | **all** | **all_local** | **all_remote**

Device

Indicates the device name of the file system for which information is displayed. File system names do not need to be fully qualified. **fs0** is as acceptable as **/dev/fs0**.

all Indicates all file systems that are known to this cluster.

all_local

Indicates all file systems that are owned by this cluster.

all_remote

Indicates all file systems that are owned by another cluster.

This must be the first parameter.

Options

- A** Automatically mounts the file system when the GPFS daemon starts.
- a** Displays the estimated average file size, in bytes.
- B** Displays the size of the data block, in bytes.
- D** Displays the type of file locking semantics that are in effect (**nfs4** or **posix**).
- d** Displays the names of all of the disks in the file system.
- E** Displays the exact **mtime** values reported.
- F** Displays the maximum number of files that are currently supported.
- f** Displays the minimum fragment size, in bytes.
- I** Displays the indirect block size, in bytes.
- i** Displays the inode size, in bytes.
- j** Displays the block allocation type.

- k** Displays the type of authorization supported by the file system.
- K** Displays the strict replication enforcement.
- L** Displays the internal log file size.
- M** Displays the maximum number of metadata replicas.
- m** Displays the default number of metadata replicas.
- n** Displays the estimated number of nodes for mounting the file system.
- o** Displays the additional mount options.
- P** Displays the storage pools defined within the file system.
- Q** Displays which quotas are currently enforced on the file system.
- R** Displays the maximum number of data replicas.
- r** Displays the default number of data replicas.
- S** Displays whether the updating of **atime** is suppressed for the **gpfs_stat()**, **gpfs_fstat()**, **stat()**, and **fstat()** calls.
- t** Displays the Windows drive letter.
- T** Displays the default mount point.
- u** Displays whether support for large LUNs is enabled.
- V** Displays the current format version of the file system.
- z** Displays whether DMAPI is enabled for this file system.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user:

1. You may issue the **mmisfs** command from any node in the GPFS cluster.
2. When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:
 - a. Proper authorization is granted to all nodes in the GPFS cluster.
 - b. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.
3. As root, a user can also issue the **mmisfs** on remote file systems.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the **mmisfs** command was issued.

Examples

If you issue the **mmisfs** command with no options for the file system **gpfs2**:

```
mmisfs gpfs2
```

The system displays information similar to this (output appears in the order that the options were specified in the command):

flag	value	description
-f	512	Minimum fragment size in bytes
-i	512	Inode size in bytes
-I	8192	Indirect block size in bytes
-m	2	Default number of metadata replicas
-M	2	Maximum number of metadata replicas
-r	1	Default number of data replicas
-R	2	Maximum number of data replicas
-j	cluster	Block allocation type
-D	posix	File locking semantics in effect
-k	all	ACL semantics in effect
-a	-1	Estimated average file size
-n	1000	Estimated number of nodes that will mount file system
-B	16384	Block size
-Q	user;group;fileset	Quotas enforced
	none	Default quotas enabled
-F	3000096	Maximum number of inodes
-V	10.00 (3.2.0.0)	File system version. Highest supported version: 10.00
-u	yes	Support for large LUNs?
-z	no	Is DMAPI enabled?
-L	524288	Logfile size
-E	yes	Exact mtime mount option
-S	no	Suppress atime mount option
-K	whenpossible	Strict replica allocation option
-P	system	Disk storage pools in file system
-d	hd3n97;hd4n97;hd5n98;hd6n98;hd7vsdn97;hd8vsdn97	Disks in file system
-A	no	Automatic mount option
-o	none	Additional mount options
-T	/gpfs2	Default mount point
-t	U	Default Windows drive letter

If you issue the **mmfsfs** command with the **all** option:

```
mmfsfs all -A
```

The system displays information similar to:

File system attributes for /dev/fs1:

=====

flag	value	description
-A	yes	Automatic mount option

File system attributes for /dev/fs2:

=====

flag	value	description
-A	yes	Automatic mount option

File system attributes for /dev/fs3:

=====

flag	value	description
-A	no	Automatic mount option

See also

“mmcrfs Command” on page 130

“mmchfs Command” on page 109

“mmdelfs Command” on page 168

Location

`/usr/lpp/mmfs/bin`

mmlsmgr Command

Displays which node is the file system manager for the specified file systems.

Synopsis

mmlsmgr [*Device* [*Device...*]]

Or,

mmlsmgr -C *ClusterName*

Or,

mmlsmgr -c

Description

Use the **mmlsmgr** command to display which node is the file system manager or cluster manager for the file system.

If you do not provide a *Device* operand, file system managers for all file systems within the current cluster for which a file system manager has been appointed are displayed.

Parameters

Device

The device names of the file systems for which the file system manager information is displayed.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

If no file system is specified, information about all file systems is displayed.

-C *ClusterName*

Displays the name of the nodes that are file system managers in cluster *ClusterName*.

-c Displays the current cluster manager node.

Options

NONE

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user:

1. You may issue the **mmlsmgr** command from any node in the GPFS cluster.
2. When using the **rsh** and **rlogin** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:
 - a. Proper authorization is granted to all nodes in the GPFS cluster.

- b. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.
3. As root, a user can also issue the **mmismgr** on remote file systems.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the **mmismgr** command was issued.

Examples

1. To display the file system manager node information for all the file systems, issue this command:

```
mmismgr
```

The system displays information similar to:

file system	manager node
fs3	9.114.94.65 (c154n01)
fs2	9.114.94.73 (c154n09)
fs1	9.114.94.81 (c155n01)

Cluster manager node: 9.114.94.65 (c154n01)

The output shows the device name of the file system and the file system manager's node number and name, in parenthesis, as they are recorded in the GPFS cluster data.

2. To display the file system manager information for file systems **gpfs2** and **gpfs3**, issue this command:

```
mmismgr gpfs2 gpfs3
```

The system displays information similar to:

file system	manager node [from 199.116.68.69 (k156gn02)]
gpfs2	199.116.68.70 (k154gn02)
gpfs3	199.116.68.72 (kolt2g_r1b42)

See also

"mmchmgr Command" on page 114

Location

/usr/lpp/mmfs/bin

mmlsmount Command

Lists the nodes that have a given GPFS file system mounted.

Synopsis

```
mmlsmount {Device | all | all_local | all_remote} [-L ] [-C {all | all_remote | ClusterName[,ClusterName...]} ]
```

Description

The **mmlsmount** command reports if a file system is in use at the time the command is issued. A file system is considered to be in use if it is explicitly mounted with the **mount** or **mmm mount** command, or if it is mounted internally for the purposes of running some other GPFS command. For example, when you run the **mmrestripefs** command, the file system will be internally mounted for the duration of the command. If **mmlsmount** is issued in the interim, the file system will be reported as being in use by the **mmlsmount** command but, unless it is explicitly mounted, will not show up in the output of the **mount** or **df** commands.

Parameters

Device | **all** | **all_local** | **all_remote**

Device

Indicates the device name of the file system for which information is displayed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

all Indicates all file systems known to this cluster.

all_local

Indicates all file systems owned by this cluster.

all_remote

Indicates all file systems owned by another cluster.

This must be the first parameter.

Options

-C {**all** | **all_remote** | *ClusterName*[,*ClusterName*...]}

Specifies the clusters for which mount information is requested. If one or more *ClusterName* is specified, only the names of nodes that belong to these clusters and have the file system mounted are displayed. The dot character (".") can be used in place of the cluster name to denote the local cluster.

Option **-C all_remote** denotes all clusters other than the one from which the command was issued.

Option **-C all** refers to all clusters, local and remote, that can have the file system mounted. Option **-C all** is the default.

-L Specifies to list the nodes that have the file system mounted.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user, you may issue the **mmlsmount** command from any node in the GPFS cluster.

If you are a non-root user, you may specify only file systems that belong to the same cluster as the node on which the **mmismount** command was issued.

Examples

1. To see how many nodes have file system **fs2** mounted, issue this command:

```
mmismount fs2
```

Output is similar to this:

File system fs2 is mounted on 3 nodes.

2. To display all mounted file systems:

```
mmismount all
```

Output is similar to this:

File system fs1 is mounted on 17 nodes.

File system remotefs1 (remote.cluster:fs1)
is mounted on 17 nodes.

3. To display all remotely mounted file systems:

```
mmismount all_remote
```

Output is similar to this:

File system remotefs1 (remote.cluster:fs1)
is mounted on 17 nodes.

4. To list the nodes having all file systems mounted:

```
mmismount all -L
```

Output is similar to this:

File system fs1 is mounted on 7 nodes:

119.124.94.69 k154n05 dq.cluster

119.124.94.87 k155n07 dq.cluster

119.124.94.102 k156n06 remote.cluster

119.124.94.82 k155n02 remote.cluster

119.124.94.86 k155n06 dq.cluster

119.124.94.89 k155n09 dq.cluster

119.124.94.81 k155n01 remote.cluster

File system remotefs1 (remote.cluster:fs1)
is mounted on 4 nodes:

119.124.94.115 kolt1_r1b42 remote.cluster

119.124.94.65 k154n01 remote.cluster

119.124.94.69 k154n05 dq.cluster

119.124.94.73 k154n09 remote.cluster

See also

“mmmount Command” on page 237

“mmumount Command” on page 285

Location

/usr/lpp/mmfs/bin

mmlnsd Command

Displays the current Network Shared Disk (NSD) information in the GPFS cluster.

Synopsis

mmlnsd [**-a** | **-F** | **-f** *Device* | **-d** "*DiskName*;*DiskName*..."] [**-L** | **-m** | **-M** | **-X**] [**-v**]

Description

Use the **mmlnsd** command to display the current information for the NSDs belonging to the GPFS cluster. The default is to display information for all NSDs defined to the cluster (**-a**). Otherwise, you may choose to display the information for a particular file system (**-f**) or for all disks that do not belong to any file system (**-F**).

Parameters

- a** Display information for all of the NSDs belonging to the GPFS cluster. This is the default.
- f** *Device*
The device name of the file system for which you want NSD information displayed. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.
- F** Display the NSDs that do not belong to any file system in the GPFS cluster.
- d** *DiskName*;*DiskName*...]
The name of the NSDs for which you want information displayed. When you enter multiple *DiskNames*, you must separate them with semicolons and enclose the entire string of disk names in quotation marks:
"gpfs3nsd;gpfs4nsd;gpfs5nsd"

Options

- L** Displays the information in a long format that shows the NSD identifier.
- m** Maps the NSD name to its disk device name in **/dev** on the local node and, if applicable, on the NSD server nodes.
- M** Maps the NSD names to its disk device name in **/dev** on all nodes.
This is a slow operation and its usage is suggested for problem determination only.
- v** Specifies that the output should contain error information, where available.
- X** Maps the NSD name to its disk device name in **/dev** on the local node and, if applicable, on the NSD server nodes. The **-X** option also displays extended information for the NSD volume ID and information such as NSD server status and Persistent Reserve (PR) enablement in the Remarks field. Using the **-X** option is a slow operation and is recommended only for problem determination.

Exit status

- 0** Successful completion.
- nonzero**
A failure has occurred.

Security

You must have root authority to issue the **mmlnsd** command.

You may issue the **mmlnsd** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To display the default information for all of the NSDs belonging to the cluster, issue this command:

```
mmfnsnsd
```

The system displays information similar to:

File system	Disk name	NSD servers
fs2	hd3n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd4n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd5n98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd6n98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd7vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd8vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd9vsdn97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd10vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com,c5n99g.ppd.pok.ibm.com
fs2	hd11vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	hd12vsdn98	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	sdbnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdcnnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sddnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdgnnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdfnnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdhnsd	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
(free disk)	hd2n97	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com

2. To display all of the NSDs attached to the node from which the command is issued, issue this command:

```
mmfnsnsd -m
```

The system displays information similar to:

Disk name	NSD volume ID	Device	Node name	Remarks
hd10vsdn98	0972846245C8E93C	/dev/hd10vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd10vsdn98	0972846245C8E93C	/dev/hd10vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd11vsdn98	0972846245C8E93F	/dev/hd11vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd11vsdn98	0972846245C8E93F	/dev/hd11vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd12vsdn98	0972846245C8E941	/dev/hd12vsdn98	c5n97g.ppd.pok.ibm.com	server node
hd12vsdn98	0972846245C8E941	/dev/hd12vsdn98	c5n98g.ppd.pok.ibm.com	server node
hd2n97	0972846145C8E924	/dev/hdisk2	c5n97g.ppd.pok.ibm.com	server node
hd2n97	0972846145C8E924	/dev/hdisk2	c5n98g.ppd.pok.ibm.com	server node
hd3n97	0972846145C8E927	/dev/hdisk3	c5n97g.ppd.pok.ibm.com	server node
hd3n97	0972846145C8E927	/dev/hdisk3	c5n98g.ppd.pok.ibm.com	server node
hd4n97	0972846145C8E92A	/dev/hdisk4	c5n97g.ppd.pok.ibm.com	server node
hd4n97	0972846145C8E92A	/dev/hdisk4	c5n98g.ppd.pok.ibm.com	server node
hd5n98	0972846245EB501C	/dev/hdisk5	c5n97g.ppd.pok.ibm.com	server node
hd5n98	0972846245EB501C	/dev/hdisk5	c5n98g.ppd.pok.ibm.com	server node
hd6n98	0972846245DB3AD8	/dev/hdisk6	c5n97g.ppd.pok.ibm.com	server node
hd6n98	0972846245DB3AD8	/dev/hdisk6	c5n98g.ppd.pok.ibm.com	server node
hd7vsdn97	0972846145C8E934	/dev/hd7vsdn97	c5n97g.ppd.pok.ibm.com	server node

3. To display all of the NSDs in the GPFS cluster in extended format, issue this command:

```
mmfnsnsd -L
```

The system displays information similar to:

File system	Disk name	NSD volume ID	NSD servers
fs2	hd3n97	0972846145C8E927	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com
fs2	hd4n97	0972846145C8E92A	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com
fs2	hd5n98	0972846245EB501C	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	hd6n98	0972846245DB3AD8	c5n98g.ppd.pok.ibm.com,c5n97g.ppd.pok.ibm.com
fs2	sdbnsd	0972845E45C8E8ED	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdcnscd	0972845E45C8E8F6	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sddnsd	0972845E45F83FDB	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdensd	0972845E45C8E909	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdgnscd	0972845E45C8E912	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdfnsd	0972845E45F02E81	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
fs2	sdhnsd	0972845E45C8E91C	c5n94g.ppd.pok.ibm.com,c5n96g.ppd.pok.ibm.com
gpfs1	hd2n97	0972846145C8E924	c5n97g.ppd.pok.ibm.com,c5n98g.ppd.pok.ibm.com

4. To display extended disk information about disks **hd3n97**, **sdfnsd**, and **hd5n98**, issue this command:

```
mmfnsd -X -d "hd3n97;sdfnsd;hd5n98"
```

The system displays information similar to:

Disk name	NSD volume ID	Device	Devtype	Node name	Remarks
hd3n97	0972846145C8E927	/dev/hdisk3	hdisk	c5n97g.ppd.pok.ibm.com	server node,pr=no
hd3n97	0972846145C8E927	/dev/hdisk3	hdisk	c5n98g.ppd.pok.ibm.com	server node,pr=no
hd5n98	0972846245EB501C	/dev/hdisk5	hdisk	c5n97g.ppd.pok.ibm.com	server node,pr=no
hd5n98	0972846245EB501C	/dev/hdisk5	hdisk	c5n98g.ppd.pok.ibm.com	server node,pr=no
sdfnsd	0972845E45F02E81	/dev/sdf	generic	c5n94g.ppd.pok.ibm.com	server node
sdfnsd	0972845E45F02E81	/dev/sdm	generic	c5n96g.ppd.pok.ibm.com	server node

See also

“mmcrnsd Command” on page 137

“mmdelnsd Command” on page 172

Location

/usr/lpp/mmfs/bin

mmlspolicy Command

Displays policy information.

Synopsis

mmlspolicy *Device* [-L]

Description

The **mmlspolicy** command displays policy information for a given file system. This information is displayed:

- When the policy file was installed.
- The user who installed the policy file.
- The node on which the policy file was installed.
- The first line of the original policy file.

For information on GPFS policies, see the chapter *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system for which policy information is to be displayed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- L** Displays the entire original policy file. If this flag is not specified, only the first line of the original policy file is displayed

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You may issue the **mmlspolicy** command from any node in the GPFS cluster.

Examples

1. This command displays basic information for the policy installed for file system **fs2**:

```
mmlspolicy fs2
```

The system displays output similar to:

```
Policy file for file system '/dev/fs2':
  Installed by root@k155n11.kgn.ibm.com on
  Mon Dec 12 21:54:55 2005.
  First line from original file 'policyfile' was:
/* This is the policy for the fs2 GPFS file system. */
```

2. This command displays extended information for the policy installed for file system **fs2**:

```
mmlspolicy fs2 -L
```

The system displays output similar to:


```

/* This is the policy for the fs2 GPFS file system. */

/* File Placement Rules */
RULE SET POOL 'sp4' WHERE name like '%sp4%'
RULE SET POOL 'sp5' WHERE name like '%sp5%'
RULE 'default' SET POOL 'system'

/* Exclude Rule */
RULE 'Exclude root users files' EXCLUDE WHERE USER_ID = 0 AND
name like '%org%'

/* Delete Rule */
RULE 'delete files' DELETE WHERE PATH_NAME like '%tmp%'

/* Migrate Rule */
RULE 'sp4.files' MIGRATE FROM POOL 'sp4' TO POOL 'sp5' WHERE
name like '%sp4%'

/* End of Policy */

```

See also

“mmapplypolicy Command” on page 73

“mmchpolicy Command” on page 122

Location

/usr/lpp/mmfs/bin

mmlsquota Command

Displays quota information for a user, group, or fileset.

Synopsis

```
mmlsquota [-u User | -g Group | -j Fileset] [-v | -q] [-e] [-C ClusterName] [Device [Device...]]
```

Or,

```
mmlsquota -d {-u | -g | -j} [-C ClusterName] [Device [Device...]]
```

Description

For the specified *User*, *Group*, or *Fileset* the **mmlsquota** command displays information about quota limits and current usage on each file system in the cluster. This information is displayed only if quota limits have been established and the user has consumed some amount of storage. If you want quota information for a *User*, *Group*, or *Fileset* that has no file system storage allocated at the present time, you must specify the **-v** option.

If none of: **-g**, **-u**, or **-j** is specified, the default is to display only user quotas for the user who issues the command.

For each file system in the cluster, the **mmlsquota** command displays:

1. Block limits:
 - quota type (USR or GRP or FILESET)
 - current usage in KB
 - soft limit in KB
 - hard limit in KB
 - space in doubt
 - grace period
2. File limits:
 - current number of files
 - soft limit
 - hard limit
 - files in doubt
 - grace period

Because the sum of the *in-doubt* value and the current usage may not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset may be constrained by the in-doubt value. If the in-doubt value approaches a significant percentage of the quota, run the **mmcheckquota** command to account for the lost space and files.

GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmlsquota** command and the **mmrepquota** command are double the value reported by the **ls** command.

When issuing the **mmlsquota** command on a mounted file system, negative in-doubt values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

| This command cannot be run from a Windows node.

Parameters

-C *ClusterName*

Specify the name of the cluster from which the quota information is obtained (from the file systems within that cluster). If this option is omitted, the local cluster is assumed.

Device

Specifies the device name of the file system for which quota information is to be displayed. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

Options

-d Display the default quota limits for user, group, or fileset quotas.

-e Specifies that **mmfsquota** is to collect updated quota usage data from all nodes before displaying results. If this option is not specified, there is the potential to display negative usage values as the quota server may process a combination of up-to-date and back-level information.

-g *Group*

Display quota information for the user group or group ID specified in the *Group* parameter.

-j *Fileset*

Display quota information for the named fileset.

-q Prints a terse message containing information only about file systems with usage over quota.

-u *User*

Display quota information for the user name or user ID specified in the *User* parameter.

-v Display quota information on file systems where the *User*, *Group* or *Fileset* limit has been set, but the storage has not been allocated.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user, you may view quota information for all users, groups, and filesets.

If you are a non-root user, you may view only fileset quota information, your own quota information, and quota information for any groups to which you belong.

You must be a root user to use the **-d** option.

GPFS must be running on the node from which the **mmfsquota** command is issued.

Examples

Userid **paul** issued this command:

```
mmfsquota
```

The system displays information similar to:

Filesystem type	Block Limits					File Limits				
	KB	quota	limit	in_doubt	grace	files	quota	limit	in_doubt	grace
fsn	USR	728	100096	200192	4880	none	35	30	50	10 6days

This output shows the quotas for user **paul** in file system **fsn** set to a soft limit of 100096 KB, and a hard limit of 200192 KB. 728 KB is currently allocated to **paul**. 4880 KB is also in doubt, meaning that the quota system has not yet been updated as to whether this space has been used by the nodes, or whether it is still available. No grace period appears because the user has not exceeded his quota. If the user had exceeded the soft limit, the grace period would be set and the user would have that amount of time to bring his usage below the quota values. If the user failed to do so, the user would not be allocated any more space.

The soft limit for files (inodes) is set at 30 and the hard limit is 50. 35 files are currently allocated to this user, and the quota system does not yet know whether the 10 in doubt have been used or are still available. A grace period of six days appears because the user has exceeded his quota. The user would have this amount of time to bring his usage below the quota values. If the user fails to do so, the user is not allocated any more space.

See also

“mmcheckquota Command” on page 104

“mmdefquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmrepquota Command” on page 259

“mmquotaon Command” on page 251

“mmquotaoff Command” on page 249

Location

/usr/lpp/mmfs/bin

mmlssnapshot Command

Displays GPFS snapshot information for the specified file system.

Synopsis

mmlssnapshot *Device* [-d] [-Q]

Description

Use the **mmlssnapshot** command to display GPFS snapshot information for the specified file system. You may optionally display the amount of storage used by the snapshot and if quotas were set for automatic activation upon mounting of the file system at the time the snapshot was taken.

Parameters

Device

The device name of the file system for which snapshot information is to be displayed. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Options

- d** Display the amount of storage used by the snapshot.
- Q** Display whether quotas were set to be automatically activated upon mounting of the file system at the time the snapshot was taken.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must be a root user to use the **-d** and **-Q** options.

If you are a root user, you can issue the **mmlssnapshot** command from any node in the GPFS cluster.

If you are a non-root user, you may only specify file systems that belong to the same cluster as the node on which the **mmlssnapshot** command was issued.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To display the snapshot information for the file system **fs1** additionally requesting storage information, issue this command:

```
mmlssnapshot fs1 -d
```

The system displays information similar to:

Snapshots in file system fs1: [data and metadata in KB]

Directory	SnapId	Status	Created	Data	Metadata
snap1	1	Valid	Fri Oct 17 10:56:22 2003	0	512

See also

“mmcrsnapshot Command” on page 142

“mmdelsnapshot Command” on page 174

“mmrestorefs Command” on page 262

“mmsnapdir Command” on page 277

Location

/usr/lpp/mmfs/bin

mmmount Command

Mounts GPFS file systems on one or more nodes in the cluster.

Synopsis

- | **mmmount** {*Device* | *DefaultMountPoint* | *DefaultDriveLetter* | **all** | **all_local** | **all_remote**} [-o *MountOptions*] [-a | -N {*Node*[,*Node*...]} | *NodeFile* | *NodeClass*}]
- Or,
- | **mmmount** *Device* {*MountPoint* | *DriveLetter*} [-o *MountOptions*] [-a | -N {*Node*[,*Node*...]} | *NodeFile* | *NodeClass*}]

Description

The **mmmount** command mounts the specified GPFS file system on one or more nodes in the cluster. If no nodes are specified, the file systems are mounted only on the node from which the command was issued. A file system can be specified using its device name or its default mount point, as established by the **mmcrfs**, **mmchfs** or **mmremotefs** commands.

When **all** is specified in place of a file system name, all GPFS file systems will be mounted. This also includes remote GPFS file systems to which this cluster has access.

Parameters

Device | *DefaultMountPoint* | *DefaultDriveLetter* | **all** | **all_local** | **all_remote**

Indicates the file system or file systems to be mounted.

Device

The device name of the file system to be mounted. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

DefaultMountPoint

The mount point associated with the file system as a result of the **mmcrfs**, **mmchfs**, or **mmremotefs** commands.

| *DefaultDriveLetter*

| The Windows drive letter associated with the file system as a result of the **mmcrfs** or
| **mmchfs** command.

all Indicates all file systems known to this cluster.

all_local
Indicates all file systems owned by this cluster.

all_remote
Indicates all files systems owned by another cluster to which this cluster has access.

This must be the first parameter.

| *DriveLetter*

| The location where the file system is to be mounted. If not specified, the file system is mounted at
| its default drive letter. This option can be used to mount a file system at a drive letter other than its
| default one or to mount a file system that does not have an established default drive letter.

MountPoint

The location where the file system is to be mounted. If not specified, the file system is mounted at its default mount point. This option can be used to mount a file system at a mount point other than its default mount point.

Options

-a Mount the file system on all nodes in the GPFS cluster.

-N {Node[,Node...] | NodeFile | NodeClass}

Specifies the nodes on which the file system is to be mounted. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

-o MountOptions

Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see “GPFS-specific mount options” on page 17.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmmount** command.

You may issue the **mmmount** command from any node in the GPFS cluster.

Examples

1. To mount all GPFS file systems on all of the nodes in the cluster, issue this command:

```
mmmount all -a
```

2. To mount file system **fs2** read-only on the local node, issue this command:

```
mmmount fs2 -o ro
```

3. To mount file system **fs1** on all NSD server nodes, issue this command:

```
mmmount fs1 -N nsdsnodes
```

See also

“mmumount Command” on page 285

“mmlsmount Command” on page 225

Location

/usr/lpp/mmfs/bin

mmnsddiscover Command

Rediscovered paths to the specified network shared disks on the specified nodes.

Synopsis

mmnsddiscover [**-a** | **-d** "*Disk[:Disk...]*" | **-F DiskFile**] [**-C ClusterName**] [**-N** {*Node[,Node...]* | *NodeFile* | *NodeClass*}]

Description

The **mmnsddiscover** command is used to rediscover paths for GPFS NSDs on one or more nodes. If you do not specify a node, GPFS rediscovered NSD paths on the node from which you issued the command.

On server nodes, **mmnsddiscover** causes GPFS to rediscover access to disks, thus restoring paths which may have been broken at an earlier time. On client nodes, **mmnsddiscover** causes GPFS to refresh its choice of which NSD server to use when an I/O operation occurs.

| In general, after the path to a disk is fixed, the **mmnsddiscover** command must be first run on the server
| that lost the path to the NSD. After that, run the command on all client nodes that need to access the NSD
| on that server. You can achieve the same effect with a single **mmnsddiscover** invocation if you utilize the
| **-N** option to specify a node list that contains all the NSD servers and clients that need to rediscover paths.

Parameters

-a Rediscovered paths for all NSDs. This is the default.

-d "*DiskName[:DiskName]*"
Specifies a list of NSDs whose paths are to be rediscovered.

-F DiskFile
Specifies a file that contains the names of the NSDs whose paths are to be rediscovered.

-C ClusterName
Specifies the name of the cluster to which the NSDs belong. This defaults to the local cluster if not specified.

-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}
Specifies the nodes on which the rediscovery is to be done.

For general information on how to specify node names, see "Specifying nodes as input to GPFS commands" on page 2.

Options

None

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmnsddiscover** command.

You can issue the **mmnsddiscover** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly-configured **.rhosts** file must exist in the home directory for the root user on each node in the GPFS cluster. If you have designated the use of a different remote communication program using the **mmcrcluster** command or the **mmchcluster** command, you must make sure that:

- Proper authorization is granted to all nodes in the GPFS cluster.
- The nodes in the GPFS cluster can communicate without the use of a password and without any extraneous messages.

Examples

1. To rediscover the paths for all of the NSDs in the local cluster on the local node, issue the command:

```
mmnsddiscover
```

The system displays output similar to:

```
mmnsddiscover: Attempting to rediscover the disks.  
This may take a while ...
```

```
mmnsddiscover: Finished.
```

2. To rediscover the paths for all of the NSDs in the local cluster on all nodes in the local cluster, issue the command:

```
mmnsddiscover -a -N all
```

The system displays output similar to:

```
mmnsddiscover: Attempting to rediscover the disks.  
This may take a while ...
```

```
mmnsddiscover: Finished.
```

See also

“mmchnsd Command” on page 120

“mmcrnsd Command” on page 137

“mmdelnsd Command” on page 172

“mmlnsd Command” on page 227

Location

/usr/lpp/mmfs/bin

mmpmon Command

Manages performance monitoring and displays performance information.

Synopsis

mmpmon [-i *CommandFile*] [-d *IntegerDelayValue*] [-p] [-r *IntegerRepeatValue*] [-s] [-t *IntegerTimeoutValue*]

Description

Before attempting to use **mmpmon**, IBM suggests that you review this command entry, then read the entire topic, *Monitoring GPFS I/O performance with the mmpmon command* in *General Parallel File System: Advanced Administration Guide*.

Use the **mmpmon** command to manage GPFS performance monitoring functions and display performance monitoring data. The **mmpmon** command reads requests from an input file or standard input (stdin), and writes responses to standard output (stdout). Error messages go to standard error (stderr). Prompts, if not suppressed, go to stderr.

When running **mmpmon** in such a way that it continually reads input from a pipe (the driving script or application never intends to send an end-of-file to **mmpmon**), set the **-r** option value to 1 (or use the default value of 1) to prevent **mmpmon** from caching the input records. This avoids unnecessary memory consumption.

| This command cannot be run from a Windows node.

Results

The performance monitoring request is sent to the GPFS daemon running on the same node that is running the **mmpmon** command.

All results from the request are written to stdout.

There are two output formats:

- Human readable, intended for direct viewing.
In this format, the results are keywords that describe the value presented, followed by the value. For example:
disks: 2
- Machine readable, an easily parsed format intended for further analysis by scripts or applications.
In this format, the results are strings with values presented as keyword/value pairs. The keywords are delimited by underscores (_) and blanks to make them easier to locate.

For details on how to interpret the **mmpmon** command results, see the topic *Monitoring GPFS I/O performance with the mmpmon command* in *General Parallel File System: Advanced Administration Guide*.

Parameters

-i *CommandFile*

The input file contains **mmpmon** command requests, one per line. Use of the **-i** flag implies use of the **-s** flag. For interactive use, just omit the **-i** flag. In this case, the input is then read from stdin, allowing **mmpmon** to take keyboard input or output piped from a user script or application program.

Leading blanks in the input file are ignored. A line beginning with a pound sign (#) is treated as a comment. Leading blanks in a line whose first non-blank character is a pound sign (#) are ignored.

Input requests to the **mmpmon** command are:

fs_io_s

Displays I/O statistics per mounted file system

io_s Displays I/O statistics for the entire node

nlist add *name[name...]*

Adds node names to a list of nodes for **mmpmon** processing

nlist del

Deletes a node list

nlist new *name[name...]*

Creates a new node list

nlist s Shows the contents of the current node list.

nlist sub *name[name...]*

Deletes node names from a list of nodes for **mmpmon** processing.

once *request*

Indicates that the request is to be performed only once.

reset Resets statistics to zero.

rhist nr

Changes the request histogram facility request size and latency ranges.

rhist off

Disables the request histogram facility. This is the default.

rhist on

Enables the request histogram facility.

rhist p

Displays the request histogram facility pattern.

rhist reset

Resets the request histogram facility data to zero.

rhist s

Displays the request histogram facility statistics values.

ver Displays **mmpmon** version.

Options

-d *IntegerDelayValue*

Specifies a number of milliseconds to sleep after one invocation of all the requests in the input file. The default value is 1000. This value must be an integer greater than or equal to 500 and less than or equal to 8000000.

The input file is processed as follows: The first request is processed, it is sent to the GPFS daemon, the responses for this request are received and processed, the results for this request are displayed, and then the next request is processed and so forth. When all requests from the input file have been processed once, the **mmpmon** command sleeps for the specified number of milliseconds. When this time elapses, **mmpmon** wakes up and processes the input file again, depending on the value of the **-r** flag.

-p Indicates to generate output that can be parsed by a script or program. If this option is not specified, human readable output is produced.

-r *IntegerRepeatValue*

Specifies the number of times to run all the requests in the input file.

The default value is one. Specify an integer between zero and 8000000. Zero means to run forever, in which case processing continues until it is interrupted. This feature is used, for example, by a driving script or application program that repeatedly reads the result from a pipe.

The **once** prefix directive can be used to override the **-r** flag. See the description of **once** in *Monitoring GPFS I/O performance with the mmpmon command* in *General Parallel File System: Advanced Administration Guide*.

-s Indicates to suppress the prompt on input.

Use of the **-i** flag implies use of the **-s** flag. For use in a pipe or with redirected input (<), the **-s** flag is preferred. If not suppressed, the prompts go to standard error (stderr).

-t IntegerTimeoutValue

Specifies a number of seconds to wait for responses from the GPFS daemon before considering the connection to have failed.

The default value is 60. This value must be an integer greater than or equal to 1 and less than or equal to 8000000.

Exit status

- 0 Successful completion.
- 1 Various errors (insufficient memory, input file not found, incorrect option, and so forth).
- 3 Either no commands were entered interactively, or there were no **mmpmon** commands in the input file. The input file was empty, or consisted of all blanks or comments.
- 4 **mmpmon** terminated due to a request that was not valid.
- 5 An internal error has occurred.
- 111 An internal error has occurred. A message will follow.

Restrictions

1. Up to five instances of **mmpmon** may be run on a given node concurrently. However, concurrent users may interfere with each other. See *Monitoring GPFS I/O performance with the mmpmon command* in *General Parallel File System: Advanced Administration Guide*.
2. Do not alter the input file while **mmpmon** is running.
3. The input file must contain valid input requests, one per line. When an incorrect request is detected by **mmpmon**, it issues an error message and terminates. Input requests that appear in the input file before the first incorrect request are processed by **mmpmon**.

Security

The **mmpmon** command must be run by a user with root authority, on the node for which statistics are desired.

Examples

1. Assume that **infile** contains these requests:

```
ver
io_s
fs_io_s
rhist off
```

and this command is issued:

```
mmpmon -i infile -r 10 -d 5000
```

The output (sent to stdout) is similar to this:

```

mmpmon node 192.168.1.8 name node1 version 3.1.0
mmpmon node 192.168.1.8 name node1 io_s OK
timestamp:      1083350358/935524
bytes read:      0
bytes written:   0
opens:           0
closes:          0
reads:           0
writes:          0
readdir:         0
inode updates:   0
mmpmon node 192.168.1.8 name node1 fs_io_s status 1
no file systems mounted
mmpmon node 192.168.1.8 name node1 rhist off OK

```

The requests in the input file are run 10 times, with a delay of 5000 milliseconds (5 seconds) between invocations.

2. Here is the previous example with the **-p** flag:

```
mmpmon -i infile -p -r 10 -d 5000
```

The output (sent to stdout) is similar to this:

```

_ver_n_192.168.1.8_nn_node1_v_2_lv_3_vt_0
_io_s_n_192.168.1.8_nn_node1_rc_0_t_1084195701_tu_350714_br_0_bw_0_oc_0
_cc_0_rdc_0_wc_0_dir_0_iu_0
_fs_io_s_n_192.168.1.8_nn_node1_rc_1_t_1084195701_tu_364489_cl_-_fs_-_rhist_
_n_192.168.1.8_nn_node1_req_off_rc_0_t_1084195701_tu_378217

```

3. This is an example of **fs_io_s** with a mounted file system:

```

mmpmon node 198.168.1.8 name node1 fs_io_s OK
cluster: node1.localdomain
filesystem: gpfs1
disks: 1
timestamp: 1093352136/799285
bytes read: 52428800
bytes written: 87031808
opens: 6
closes: 4
reads: 51
writes: 83
readdir: 0
inode updates: 11

```

```

mmpmon node 198.168.1.8 name node1 fs_io_s OK
cluster: node1.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1093352136/799285
bytes read: 87031808
bytes written: 52428800
opens: 4
closes: 3
reads: 12834
writes: 50
readdir: 0
inode updates: 9

```

4. Here is the previous example with the **-p** flag:

```

_fs_io_s_n_198.168.1.8_nn_node1_rc_0_t_1093352061_tu_93867_cl_node1.localdomain
_fs_gpfs1_d_1_br_52428800_bw_87031808_oc_6_cc_4_rdc_51_wc_83_dir_0_iu_10
_fs_io_s_n_198.168.1.8_nn_node1_rc_0_t_1093352061_tu_93867_cl_node1.localdomain
_fs_gpfs2_d_2_br_87031808_bw_52428800_oc_4_cc_3_rdc_12834_wc_50_dir_0_iu_8

```

This output consists of two strings.

5. This is an example of **io_s** with a mounted file system:

```
mmpmon node 198.168.1.8 name node1 io_s OK
timestamp: 1093351951/587570
bytes read: 139460608
bytes written: 139460608
opens: 10
closes: 7
reads: 12885
writes: 133
readdir: 0
inode updates: 14
```

6. Here is the previous example with the **-p** flag:

```
_io_s_ _n_ 198.168.1.8 _nn_ node1 _rc_ 0 _t_ 1093351982 _tu_ 356420 _br_ 139460608
_bw_ 139460608 _oc_ 10 _cc_ 7 _rdc_ 0 _wc_ 133 _dir_ 0 _iu_ 14
```

This output consists of one string.

For several more examples, see *Monitoring GPFS I/O performance with the mmpmon command* in *General Parallel File System: Advanced Administration Guide*.

Location

/usr/lpp/mmfs/bin

mmputacl Command

Sets the GPFS access control list for the specified file or directory.

Synopsis

mmputacl [-d] [-i *InFilename*] *Filename*

Description

Use the **mmputacl** command to set the ACL of a file or directory.

| This command cannot be run from a Windows node.

If the **-i** option is not used, the command expects the input to be supplied through standard input, and waits for your response to the prompt.

For information about NFS V4 ACLs, see Chapter 6, “Managing GPFS access control lists and NFS export,” on page 49.

Any output from the **mmgetacl** command can be used as input to **mmputacl**. The command is extended to support NFS V4 ACLs. In the case of NFS V4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual access control entries can be flagged as being inherited (either by files, directories, both, or neither). Consequently, specifying the **-d** flag for an NFS V4 ACL is an error. By its nature, storing an NFS V4 ACL implies changing the inheritable entries (the GPFS default ACL) as well.

Table 7 describes how **mmputacl** works.

Table 7. The **mmputacl** command for POSIX and NFS V4 ACLs

Command	POSIX ACL	NFS V4 ACL
mmputacl	Access ACL (Error if default ACL is NFS V4 [1])	Stores the ACL (implies default as well)
mmputacl -d	Default ACL (Error if access ACL is NFS V4 [1])	Error: NFS V4 ACL (has no default ACL)
[1] The default and access ACLs are not permitted to be mixed types because NFS V4 ACLs include inherited entries, which are the equivalent of a default ACL. An mmdelacl of the NFS V4 ACL is required before an ACL is converted back to POSIX.		

Depending on the file system’s **-k** setting (**posix**, **nfs4**, or **all**), **mmputacl** may be restricted. The **mmputacl** command is not allowed to store an NFS V4 ACL if **-k posix** is in effect. The **mmputacl** command is not allowed to store a POSIX ACL if **-k nfs4** is in effect. For more information, see the description of the **-k** flag for the **mmchfs**, **mmcrfs**, and **mmlsfs** commands.

Note that the test to see if the given ACL is acceptable based on the file system’s **-k** setting cannot be done until after the ACL is provided. For example, if **mmputacl file1** is issued (no **-i** flag specified) the user then has to input the ACL before the command can verify that it is an appropriate ACL given the file system settings. Likewise, the command **mmputacl -d dir1** (again the ACL was not given with the **-i** flag) requires that the ACL be entered before file system ACL settings can be tested. In this situation, the **-i** flag may be preferable to manually entering a long ACL, only to find out it is not allowed by the file system.

Parameters

Filename

The path name of the file or directory for which the ACL is to be set. If the **-d** option is specified, *Filename* must be the name of a directory.

Options

-d Specifies that the default ACL of a directory is to be set. This flag cannot be used on an NFS V4 ACL.

-i *InFilename*

The path name of a source file from which the ACL is to be read.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You may issue the **mmputacl** command only from a node in the GPFS cluster where the file system is mounted.

You must be the file or directory owner, the root user, or someone with control permission in the ACL, to run the **mmputacl** command.

Examples

To use the entries in a file named **standard.acl** to set the ACL for a file named **project2.history**, issue this command:

```
mmputacl -i standard.acl project2.history
```

where **standard.acl** contains:

```
user::rwx-  
group::rwx-  
other::--x-  
mask::rw-c  
user:alpha:rwx-  
group:audit:rwx-  
group:system:-w--
```

To confirm the change, issue this command:

```
mmgetacl project.history
```

The system displays information similar to:

```
#owner:paul  
#group:design  
user::rwx-  
group::rwx-  
other::--x-  
mask::rw-c  
user:alpha:rwx-  
group:audit:rwx-  
group:system:-w--
```

See also

“**mmeditACL** Command” on page 179

“**mmdelACL** Command” on page 160

“**mmgetACL** Command” on page 196

Location

`/usr/lpp/mmfs/bin`

mmquotaoff Command

Deactivates quota limit checking.

Synopsis

mmquotaoff [-u] [-g] [-j] [-v] {*Device* [*Device* ...] | -a}

Description

The **mmquotaoff** command disables quota limit checking by GPFS.

If none of: **-u**, **-j** or **-g** is specified, the **mmquotaoff** command deactivates quota limit checking for users, groups, and filesets.

If the **-a** option is not specified, *Device* must be the last parameter entered.

Parameters

Device[*Device* ...]

The device name of the file system to have quota limit checking deactivated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- a** Deactivates quota limit checking for all GPFS file systems in the cluster. When used in combination with the **-g** option, only group quota limit checking is deactivated. When used in combination with the **-u** or **-j** options, only user or fileset quota limit checking, respectively, is deactivated.
- g** Specifies that only group quota limit checking is to be deactivated.
- j** Specifies that only quota checking for filesets is to be deactivated.
- u** Specifies that only user quota limit checking is to be deactivated.
- v** Prints a message for each file system in which quotas are deactivated.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmquotaoff** command.

GPFS must be running on the node from which the **mmquotaoff** command is issued.

Examples

1. To deactivate user quota limit checking on file system **fs0**, issue this command:

```
mmquotaoff -u fs0
```

To confirm the change, issue this command:

```
mmfs fs0 -Q
```

The system displays information similar to:

flag	value	description
-Q	group;fileset	Quotas enforced

2. To deactivate group quota limit checking on all file systems, issue this command:

```
mmquotaoff -g -a
```

To confirm the change, individually for each file system, issue this command:

```
mmllsfs fs2 -Q
```

The system displays information similar to:

flag	value	description
-Q	user;fileset	Quotas enforced

3. To deactivate all quota limit checking on file system **fs0**, issue this command:

```
mmquotaoff fs0
```

To confirm the change, issue this command:

```
mmllsfs fs0 -Q
```

The system displays information similar to:

flag	value	description
-Q	none	Quotas enforced

See also

“mmcheckquota Command” on page 104

“mmdefedquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmllsquota Command” on page 232

“mmquotaon Command” on page 251

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmquotaon Command

Activates quota limit checking.

Synopsis

mmquotaon [-u] [-g] [-j] [-v] {*Device* [*Device*...] | -a}

Description

The **mmquotaon** command enables quota limit checking by GPFS.

If none of: **-u**, **-j** or **-g** is specified, the **mmquotaon** command activates quota limit checking for users, groups, and filesets.

If the **-a** option is not used, *Device* must be the last parameter specified.

After quota limit checking has been activated by issuing the **mmquotaon** command, issue the **mmcheckquota** command to count inode and space usage.

Parameters

Device[*Device* ...]

The device name of the file system to have quota limit checking activated.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- a** Activates quota limit checking for all of the GPFS file systems in the cluster. When used in combination with the **-g** option, only group quota limit checking is activated. When used in combination with the **-u** or **-j** option, only user or fileset quota limit checking, respectively, is activated.
- g** Specifies that only group quota limit checking is to be activated.
- j** Specifies that only fileset quota checking is to be activated.
- u** Specifies that only user quota limit checking is to be activated.
- v** Prints a message for each file system in which quota limit checking is activated.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmquotaon** command.

GPFS must be running on the node from which the **mmquotaon** command is issued.

Examples

1. To activate user quotas on file system **fs0**, issue this command:

```
mmquotaon -u fs0
```

To confirm the change, issue this command:

```
mmfsfs fs0 -Q
```

The system displays information similar to:

flag	value	description
-Q	user	Quotas enforced

2. To activate group quota limit checking on all file systems, issue this command:

```
mmquotaon -g -a
```

To confirm the change, individually for each file system, issue this command:

```
mmfsfs fs1 -Q
```

The system displays information similar to:

flag	value	description
-Q	group	Quotas enforced

3. To activate user, group, and fileset quota limit checking on file system **fs2**, issue this command:

```
mmquotaon fs2
```

To confirm the change, issue this command:

```
mmfsfs fs2 -Q
```

The system displays information similar to:

flag	value	description
-Q	user;group;fileset	Quotas enforced

See also

“mmcheckquota Command” on page 104

“mmdefquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmlsquota Command” on page 232

“mmquotaoff Command” on page 249

“mmrepquota Command” on page 259

Location

/usr/lpp/mmfs/bin

mmremoteclass Command

Manages the information about other GPFS clusters that this cluster can access when mounting remote GPFS file systems.

Synopsis

mmremoteclass add *RemoteClusterName* [-n *ContactNodes*] [-k *KeyFile*]

Or,

mmremoteclass update *RemoteClusterName* [-C *NewClusterName*] [-n *ContactNodes*] [-k *KeyFile*]

Or,

mmremoteclass delete {*RemoteClusterName* | all}

Or,

mmremoteclass show [*RemoteClusterName* | all]

Description

The **mmremoteclass** command is used to make remote GPFS clusters known to the local cluster, and to maintain the attributes associated with those remote clusters. The keyword appearing after **mmremoteclass** determines which action is performed:

add Adds a remote GPFS cluster to the set of remote clusters known to the local cluster.

delete Deletes the information for a remote GPFS cluster.

show Displays information about a remote GPFS cluster.

update
Updates the attributes of a remote GPFS cluster.

To be able to mount file systems that belong to some other GPFS cluster, you must first make the nodes in this cluster aware of the GPFS cluster that owns those file systems. This is accomplished with the **mmremoteclass add** command. The information that the command requires must be provided to you by the administrator of the remote GPFS cluster. You will need this information:

- The name of the remote cluster.
- The names or IP addresses of a few nodes that belong to the remote GPFS cluster.
- The public key file generated by the administrator of the remote cluster by running the **mmauth genkey** command for the remote cluster.

Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters like there is between the nodes within a cluster. This means that once a remote cluster is defined with the **mmremoteclass** command, the information about that cluster is automatically propagated across all nodes that belong to this cluster. But if the administrator of the remote cluster decides to rename it, or deletes some or all of the contact nodes, or change the public key file, the information in this cluster becomes obsolete. It is the responsibility of the administrator of the remote GPFS cluster to notify you of such changes so that you can update your information using the appropriate options of the **mmremoteclass update** command.

| Remote file system access is not supported for clusters that contain Windows nodes.

Parameters

RemoteClusterName

Specifies the cluster name associated with the remote cluster that owns the remote GPFS file system. The value **all** indicates all remote clusters defined to this cluster, when using the **mmremoteccluster delete** or **mmremoteccluster show** commands.

-C *NewClusterName*

Specifies the new cluster name to be associated with the remote cluster.

-k *KeyFile*

Specifies the name of the public key file provided to you by the administrator of the remote GPFS cluster.

-n *ContactNodes*

A comma separated list of nodes that belong to the remote GPFS cluster, in this format:

[tcpPort=NNNN,]node1[,node2 ...]

where:

tcpPort=NNNN

Specifies the TCP port number to be used by the local GPFS daemon when contacting the remote cluster. If not specified, GPFS will use the default TCP port number 1191.

node1[,node2...]

Specifies a list of nodes that belong to the remote cluster. The nodes can be identified through their host names or IP addresses.

Options

None.

Exit status

0 Successful completion. After successful completion of the **mmremoteccluster** command, the new configuration information is propagated to all nodes in the cluster.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmremoteccluster** command.

You may issue the **mmremoteccluster** command from any node in the GPFS cluster.

Examples

1. This command adds remote cluster **k164.kgn.ibm.com** to the set of remote clusters known to the local cluster, specifying **k164n02** and **k164n03** as remote contact nodes. File **k164.id_rsa.pub** is the name of the public key file provided to you by the administrator of the remote cluster.

```
mmremoteccluster add k164.kgn.ibm.com -n k164n02,k164n03\  
-k k164.id_rsa.pub
```

The output is similar to this:

```
mmremoteccluster: 6027-1371 Propagating the changes to all  
affected nodes. This is an asynchronous process.
```

2. This command displays information for the remote cluster **k164.kgn.ibm.com**.

```
mmremoteccluster show k164.kgn.ibm.com
```


The output is similar to this:

```
Cluster name:    k164.kgn.ibm.com
Contact nodes:   k164n02,k164n03
SHA digest:      a3917c8282fca7a27d951566940768dcd241902b
File systems:    (none defined)
```

For more information on the SHA digest, see *General Parallel File System: Problem Determination Guide* and search on *SHA digest*.

3. This command updates information for the remote cluster **k164.kgn.ibm.com**, changing the remote contact nodes to **k164n02** and **k164n01**. The TCP port to be used when contacting cluster **k164.kgn.ibm.com** is defined to be 6667..

```
mmremoteccluster update k164.kgn.ibm.com -n tcpPort=6667,k164n02,k164n01
```

The output is similar to this:

```
mmremoteccluster: 6027-1371 Propagating the changes to all
affected nodes. This is an asynchronous process.
```

The **mmremoteccluster show** command can then be used to see the changes.

```
mmremoteccluster show k164.kgn.ibm.com
```

The output is similar to this:

```
Cluster name:    k164.kgn.ibm.com
Contact nodes:   tcpPort=6667,k164n02,k164n01
SHA digest:      a3917c8282fca7a27d951566940768dcd241902b
File systems:    (none defined)
```

For more information on the SHA digest, see *General Parallel File System: Problem Determination Guide* and search on *SHA digest*.

4. This command deletes information for remote cluster **k164.kgn.ibm.com** from the local cluster.

```
mmremoteccluster delete k164.kgn.ibm.com
```

The output is similar to this:

```
mmremoteccluster: 6027-1371 Propagating the changes to all
affected nodes. This is an asynchronous process.
```

See also

“mmauth Command” on page 78

“mmremotefs Command” on page 256

Accessing GPFS file systems from other GPFS clusters in General Parallel File System: Advanced Administration Guide.

Location

/usr/lpp/mmfs/bin

mmremotefs Command

Manages the information about GPFS file systems from other clusters that this cluster can mount.

Synopsis

mmremotefs add *Device* **-f** *RemoteDevice* **-C** *RemoteClusterName* **[-T** *MountPoint***]** **[-A {yes | no | automount}]** **[-o** *MountOptions***]**

Or,

mmremotefs delete {*Device* | **all** | **-C** *RemoteClusterName*}

Or,

mmremotefs show [*Device* | **all** | **-C** *RemoteClusterName*]

Or,

mmremotefs update *Device* **[-f** *RemoteDevice***]** **[-C** *RemoteClusterName***]** **[-T** *MountPoint***]** **[-A {yes | no | automount}]** **[-o** *MountOptions***]**

Description

The **mmremotefs** command is used to make GPFS file systems that belong to other GPFS clusters known to the nodes in this cluster, and to maintain the attributes associated with these file systems. The keyword appearing after **mmremotefs** determines which action is performed:

add Define a new remote GPFS file system.

delete Delete the information for a remote GPFS file system.

show Display the information associated with a remote GPFS file system.

update

Update the information associated with a remote GPFS file system.

Use the **mmremotefs** command to make the nodes in this cluster aware of file systems that belong to other GPFS clusters. The cluster that owns the given file system must have already been defined with the **mmremotefcluster** command. The **mmremotefs** command is used to assign a local name under which the remote file system will be known in this cluster, the mount point where the file system is to be mounted in this cluster, and any local mount options that you may want.

Once a remote file system has been successfully defined and a local device name associated with it, you can issue normal commands using that local name, the same way you would issue them for file systems that are owned by this cluster.

- | When running the **mmremotefs** command delete and update options, the file system must be unmounted on the local cluster, however, it can be mounted elsewhere.
- | Remote file system access is not supported for clusters that contain Windows nodes.

Parameters

Device

Specifies the name by which the remote GPFS file system will be known in the cluster.

-C *RemoteClusterName*

Specifies the name of the GPFS cluster that owns the remote GPFS file system.

-f RemoteDevice

Specifies the actual name of the remote GPFS file system. This is the device name of the file system as known to the remote cluster that owns the file system.

Options

-A {yes | no | automount}

Indicates when the file system is to be mounted:

yes When the GPFS daemon starts.

no Manual mount. This is the default.

automount

When the file system is first accessed.

-o MountOptions

Specifies the mount options to pass to the mount command when mounting the file system. For a detailed description of the available mount options, see “GPFS-specific mount options” on page 17.

-T MountPoint

The local mount point directory of the remote GPFS file system. If it is not specified, the mount point will be set to *DefaultMountDir/Device*. The default value for *DefaultMountDir* is **/gpfs**, but it can be changed with the **mmchconfig** command.

Exit status

0 Successful completion. After successful completion of the **mmremotefs** command, the new configuration information is propagated to all nodes in the cluster.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmremotefs** command.

You may issue the **mmremotefs** command from any node in the GPFS cluster.

Examples

This command adds remote file system **gpfsn**, owned by remote cluster **k164.kgn.ibm.com**, to the local cluster, assigning **rgpfsn** as the local name for the file system, and **/gpfs/rgpfsn** as the local mount point.

```
mmremotefs add rgpfsn -f gpfsn -C k164.kgn.ibm.com -T /gpfs/rgpfsn
```

The output is similar to this:

```
mmremotefs: 6027-1371 Propagating the changes to all affected
                    nodes. This is an asynchronous process.
```

The **mmremotefs show** command can be used to see the changes.

```
mmremotefs show rgpfsn
```

The output is similar to this:

Local Name	Remote Name	Cluster name	Mount Point	Mount Options	Automount
rgpfsn	gpfsn	k164n.kgn.ibm.com	/gpfs/rgpfsn	rw,mtime,noatime	no

See also

“mmauth Command” on page 78

“mmremotecoluster Command” on page 253

Accessing GPFS file systems from other GPFS clusters in General Parallel File System: Advanced Administration Guide.

Location

/usr/lpp/mmfs/bin

mmrepquota Command

Reports file system user, group, and fileset quotas.

Synopsis

mmrepquota [-e] [-g] [-q] [-u] [-n] [-v] [-j] {*Device* [*Device...*] | -a}

Description

The **mmrepquota** command reports file system usage and quota information for a user, group, or fileset.

| This command cannot be run from a Windows node.

If none of **-g**, **-j**, or **-u** is specified, then user, group and fileset quotas are listed.

If the **-a** option is not specified, *Device* must be the last parameter entered.

For each file system in the cluster, the **mmrepquota** command displays:

1. Block limits:

- quota type (USR, GRP or FILESET)
- current usage in KB
- soft limit in KB
- hard limit in KB
- space in doubt
- grace period

2. File limits:

- current number of files
- soft limit
- hard limit
- files in doubt
- grace period

3. Entry Type

default on

Default quotas are enabled for this file system

default off

Default quotas are not enabled for this file system

e Explicit quotas – the quota limits have been explicitly set using the **mmedquota** command

d Default quotas – the quota limits are the default values set using the **mmdefedquota** command

i Initial quotas – default quotas were not enabled when this initial entry was established. Initial quota limits have a value of zero indicating no limit.

Because the sum of the in-doubt value and the current usage may not exceed the hard limit, the actual block space and number of files available to the user, group, or fileset may be constrained by the *in-doubt* value. If the *in-doubt* value approach a significant percentage of the quota, run the **mmcheckquota** command to account for the lost space and files.

GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either type of replication set to a value of two, the values reported on by both the **mmfsquota** command and the **mmrepquota** command are double the value reported by the **ls** command.

When issuing the **mmrepquota** command on a mounted file system, negative in-doubt values may be reported if the quota server processes a combination of up-to-date and back-level information. This is a transient situation and may be ignored.

Parameters

Device[Device...]

The device name of the file system to be listed.

If more than one file system is listed, the names must be delimited by a space. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

Options

- a** Lists quotas for all file systems in the cluster. A header line is printed automatically with this option.
- e** Specifies that the **mmrepquota** command is to collect updated quota usage data from all nodes before displaying results. If this option is not specified, there is the potential to display negative usage values as the quota server may process a combination of up-to-date and back-level information.
- g** List only group quotas.
- j** List only fileset quotas.
- n** Displays a numerical user ID.
- q** Show whether quota enforcement is active.
- u** List only user quotas.
- v** Print a header line.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmrepquota** command.

GPFS must be running on the node from which the **mmrepquota** command is issued.

Examples

- To report on user quotas for file system **fs2** and display a header line, issue this command:

```
mmrepquota -u -v fs2
```

The system displays information similar to:

```
*** Report for USR quotas on fs2
      Block Limits      |      File Limits
      in               |      in      entry
Name type KB quota limit doubt grace | files quota limit doubt grace Type
root  USR   8   0   0   0   none |   1   0   0   0   none default
                                   on
```

user2	USR	2016	256	512	0	6days	7	10	20	0	none	d
user3	USR	104	256	512	0	none	1	10	20	0	none	d
user4	USR	0	256	512	0	none	0	10	20	0	none	d
user5	USR	368	256	512	0	23hours	5	4	10	0	23hours	d
user6	USR	0	256	512	0	none	0	10	20	0	none	d
user7	USR	1024	1024	5120	4096	none	1	0	0	19	none	e

2. To report on quota enforcement for **fs2**, issue this command:

```
mmrepquota -q fs2
```

The system displays information similar to:

```
fs2: USR quota is on; default quota is on
fs2: GRP quota is on; default quota is on
fs2: FILESET quota is on; default quota is off
```

See also

“mmcheckquota Command” on page 104

“mmdefedquota Command” on page 150

“mmdefquotaoff Command” on page 152

“mmdefquotaon Command” on page 154

“mmedquota Command” on page 182

“mmlsquota Command” on page 232

“mmquotaoff Command” on page 249

“mmquotaon Command” on page 251

Location

/usr/lpp/mmfs/bin

mmrestorefs Command

Restores a file system from a GPFS snapshot.

Synopsis

mmrestorefs *Device Directory* [-c]

Description

Use the **mmrestorefs** command to restore user data and attribute files to a file system using those of the specified snapshot.

Prior to issuing the **mmrestorefs** command, you must unmount the file system from all nodes in the cluster. The file system may not be remounted until the **mmrestorefs** command has successfully completed, unless you have specified the **-c** option to force the restore to continue even in the event errors are encountered. Automatic quota activation upon mounting the file system is *not* restored by the **mmrestorefs** command. You must issue the **mmchfs -Q yes** command to restore automatic quota activation.

Snapshots are not affected by the **mmrestorefs** command. Consequently, a failure while restoring one snapshot may possibly be recovered by restoring a different snapshot.

When the **mmsnapdir -a** (add a snapshots subdirectory to all subdirectories in the file system) option is in effect, the snapshots subdirectories may no longer show the complete list of snapshots containing the parent directory, if the file system was restored from a snapshot that was not the latest. Since the root directory is contained in all snapshots, its snapshots subdirectory will always show the complete list of snapshots.

For information on how GPFS policies and snapshots interact, see *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For protection against media failures, see *General Parallel File System: Concepts, Planning, and Installation Guide* and search on *recoverability considerations*.

Parameters

Device

The device name of the file system for which the snapshot is to be created. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Directory

The snapshot with which to restore the file system.

Options

-c Continue to restore the file system in the event errors occur.

Upon completion of the **mmrestorefs -c** command, the file system is inconsistent, but can be mounted to recover data from the snapshot. If necessary, the command may be issued to recover as much data as possible. The **mmfsck** command may be run on an inconsistent file system.

After the **mmrestorefs -c** command has been issued, use the **mmfsck** command to clean up the files or directories that could not be restored.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmrestorefs** command.

You may issue the **mmrestorefs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

We have a directory structure similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If the directory **userA** is then deleted, we would have:

```
/fs1/file1

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

The directory **userB** is then created using the inode originally assigned to **userA**. We take another snapshot:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk
Quiescing all file system operations
Writing dirty data to disk again
Creating snapshot.
Resuming operations.
```

After the command is issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userB/file2b
/fs1/userB/file3b

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

```
/fs1/.snapshots/snap2/file1  
/fs1/.snapshots/snap2/userB/file2b  
/fs1/.snapshots/snap2/userB/file3b
```

If the file system is then to be restored from **snap1**:

```
mmrestorefs fs1 snap1
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1  
/fs1/userA/file2  
/fs1/userA/file3
```

```
/fs1/.snapshots/snap1/file1  
/fs1/.snapshots/snap1/userA/file2  
/fs1/.snapshots/snap1/userA/file3
```

```
/fs1/.snapshots/snap2/file1  
/fs1/.snapshots/snap2/userB/file2b  
/fs1/.snapshots/snap2/userB/file3b
```

See also

“mmcrsnapshot Command” on page 142

“mmdelsnapshot Command” on page 174

“mmlssnapshot Command” on page 235

“mmsnapdir Command” on page 277

Location

/usr/lpp/mmfs/bin

mmrestripefile Command

Performs a repair operation over the specified list of files.

Synopsis

mmrestripefile {-b | -m | -p | -r} {-F *FilenameFile* | *Filename* [*Filename*...]}

Description

The **mmrestripefile** command performs a repair operation over the specified list of files. The **-F** flag allows the user to specify a file containing the list of file names to be restriped, with one file name per line.

The **mmrestripefile** command attempts to restore the metadata or data replication factor of the file.

You must specify one of the four options (**-b**, **-m**, **-p**, or **-r**) to indicate how much file data to move.

If you do not use replication, the **-m** and **-r** options are equivalent. Their behavior differs only on replicated files. After a successful replicate (**-r** option), all suspended disks are empty. A migrate operation, using the **-m** option, leaves data on a suspended disk as long as at least one other replica of the data remains on a disk that is not suspended. Restriping a file system includes replicating it. The **-b** option performs all the operations of the **-m** and **-r** options.

| This command cannot be run from a Windows node.

Parameters

-F *FilenameFile*

Specifies a file that contains a list of names of files to be restriped, one name per line.

Filename

Specifies the names of one or more files to be restriped.

Options

- b** Rebalances a list of files across all disks that are not suspended, even if they are stopped. Although blocks are allocated on a stopped disk, they are not written to a stopped disk, nor are reads allowed from a stopped disk, until that disk is started and replicated data is copied onto it.
- m** Migrates critical data off of any suspended disk in this file system. Critical data is all data that would be lost if currently suspended disks were removed.
- p** Directs **mmrestripefile** to repair the file placement within the storage pool.

Files assigned to one storage pool, but with data in a different pool, will have their data migrated to the correct pool. These files are called ill-placed. Utilities, such as the **mmchattr** command, may change a file's storage pool assignment, but not move the data. The **mmrestripefile** command may then be invoked to migrate all of the data at once, rather than migrating each file individually. Note that the rebalance operation, specified by the **-b** option, also performs data placement on all files, whereas the placement option, specified by **-p**, rebalances only the files that it moves.
- r** Migrates all data for a list of files off of suspended disks. It also restores all specified replicated files in the file system to their designated degree of replication when a previous disk failure or removal of a disk has made some replicated data inaccessible. Use this option either immediately after a disk failure, to protect replicated data against a subsequent failure, or before taking a disk offline for maintenance, to protect replicated data against failure of another disk during the maintenance process.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmrestripefile** command.

You may issue the **mmrestripefile** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

This example illustrates restriping a file named **testfile0**. This command confirms that **testfile0** is illplaced.

```
mmfgetattr -L testfile0
```

The system displays output similar to:

```
file name:          testfile0
metadata replication: 1 max 2
data replication:    1 max 2
flags:              illplaced
storage pool name:   sp5
fileset name:        root
snapshot name:
```

To correct the problem, issue this command:

```
mmrestripefile -p testfile0
```

To confirm the change, issue this command:

```
mmfgetattr -L testfile0
```

The system displays output similar to:

```
file name:          testfile0
metadata replication: 1 max 2
data replication:    1 max 2
flags:
storage pool name:   sp5
fileset name:        root
snapshot name:
```

See also

“**mmadddisk** Command” on page 66

“**mmapplypolicy** Command” on page 73

“**mmchattr** Command” on page 86

“mmchdisk Command” on page 100

“mmdeldisk Command” on page 162

“mmrpldisk Command” on page 271

“mmrestripefs Command” on page 268

Location

/usr/lpp/mmfs/bin

mmrestripefs Command

Rebalances or restores the replication of all files in a file system.

Synopsis

mmrestripefs *Device* {-b | -m | -p | -r | -R} [-N {Node[,Node...] | NodeFile | NodeClass}] [-P PoolName]

Description

Use the **mmrestripefs** command to rebalance or restore the replication of all files in a file system. The **mmrestripefs** command moves existing file system data between different disks in the file system based on changes to the disk state made by the **mmchdisk**, **mmadddisk**, and **mmdeildisk** commands.

The **mmrestripefs** command attempts to restore the metadata or data replication of any file in the file system.

You must specify one of the five options (-b, -m, -p, -r, or -R) to indicate how much file system data to move. You can issue this command against a mounted or unmounted file system.

If you do not use replication, the -m and -r options are equivalent. Their behavior differs only on replicated files. After a successful replicate (-r option), all suspended disks are empty. A migrate operation, using the -m option, leaves data on a suspended disk as long as at least one other replica of the data remains on a disk that is not suspended. Restripping a file system includes replicating it. The -b option performs all the operations of the -m and -r options.

Consider the necessity of restripping and the current demands on the system. New data that is added to the file system is correctly striped. Restripping a large file system requires a large number of insert and delete operations and may affect system performance. Plan to perform this task when system demand is low.

Parameters

Device

The device name of the file system to be restripped. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

-N {Node[,Node...] | NodeFile | NodeClass}

Specify the nodes that participate in the restripe of the file system. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the restripe of the file system).

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

- b** Rebalances all files across all disks that are not suspended, even if they are stopped. Although blocks are allocated on a stopped disk, they are not written to a stopped disk, nor are reads allowed from a stopped disk, until that disk is started and replicated data is copied onto it. The **mmrestripefs** command rebalances and restripes the file system. Use this option to rebalance the file system after adding, changing, or deleting disks in a file system.

Note: Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

- m** Migrates all critical data off of any suspended disk in this file system. Critical data is all data that would be lost if currently suspended disks were removed.
- P PoolName**
Directs **mmrestripefs** to repair only files assigned to the specified storage pool.
- p** Directs **mmrestripefs** to repair the file placement within the storage pool.

Files assigned to one storage pool, but with data in a different pool, will have their data migrated to the correct pool. Such files are referred to as ill-placed. Utilities, such as the **mmchattr** command, may change a file's storage pool assignment, but not move the data. The **mmrestripefs** command may then be invoked to migrate all of the data at once, rather than migrating each file individually. Note that the rebalance operation, specified by the **-b** option, also performs data placement on all files, whereas the placement option, specified by **-p**, rebalances only the files that it moves.
- r** Migrates all data off suspended disks. It also restores all replicated files in the file system to their designated degree of replication when a previous disk failure or removal of a disk has made some replica data inaccessible. Use this parameter either immediately after a disk failure to protect replicated data against a subsequent failure, or before taking a disk offline for maintenance to protect replicated data against failure of another disk during the maintenance process.
- R** Changes the replication settings of each file, directory, and system metadata object so that they match the default file system settings (see the "mmchfs Command" on page 109 **-m** and **-r** options) as long as the maximum (**-M** and **-R**) settings for the object allow it. Next, it replicates or unreplicates the object as needed to match the new settings. This option can be used to replicate all of the existing files that had not been previously replicated or to unreplicate the files if replication is no longer needed or wanted.

Exit status

- 0** Successful completion.
- nonzero**
A failure has occurred.

Security

You must have root authority to run the **mmrestripefs** command.

You may issue the **mmrestripefs** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To move all critical data from any suspended disk in file system **fs0**, issue this command:

```
mmrestripefs fs0 -m
```

The system displays information similar to:

```
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
```

```
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
6 % complete on Fri Feb 10 15:45:07 2006
45 % complete on Fri Feb 10 15:48:03 2006
78 % complete on Fri Feb 10 15:49:28 2006
85 % complete on Fri Feb 10 15:49:53 2006
100 % complete on Fri Feb 10 15:53:21 2006
Scan completed successfully.
```

2. To rebalance all files in file system **fs1** across all defined, accessible disks that are not stopped or suspended, issue this command:

```
mmrestripefs fs1 -b
```

The system displays information similar to:

```
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scanning file system metadata for fslspl storage pool
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
0 % complete on Wed May 16 16:29:45 2007
1 % complete on Wed May 16 16:29:51 2007
2 % complete on Wed May 16 16:29:54 2007
6 % complete on Wed May 16 16:29:57 2007
13 % complete on Wed May 16 16:30:00 2007
23 % complete on Wed May 16 16:30:03 2007
34 % complete on Wed May 16 16:30:06 2007
42 % complete on Wed May 16 16:30:09 2007
62 % complete on Wed May 16 16:30:12 2007
82 % complete on Wed May 16 16:30:15 2007
100 % complete on Wed May 16 16:30:18 2007
Scan completed successfully.
```

See also

“**mmadddisk** Command” on page 66

“**mmapplypolicy** Command” on page 73

“**mmchattr** Command” on page 86

“**mmchdisk** Command” on page 100

“**mmchfs** Command” on page 109

“**mmdeldisk** Command” on page 162

“**mmrpldisk** Command” on page 271

“**mmrestripefile** Command” on page 265

Location

/usr/lpp/mmfs/bin

mmrpldisk Command

Replaces the specified disk.

Synopsis

mmrpldisk *Device* *DiskName* [*DiskDesc* | **-F** *DescFile*] [**-v** {yes | **no**}] [**-N** {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}]

Description

Use the **mmrpldisk** command to replace an existing disk in the GPFS file system with a new one. All data on the old disk is migrated to the new one.

To replace disks in a GPFS file system, you must first decide if you will:

1. Create new disks using the **mmcrnsd** command.
Use the rewritten disk descriptor file produced by the **mmcrnsd** command or create a new disk descriptor. When using the rewritten file, the *Disk Usage* and *Failure Group* specifications remain the same as specified on the **mmcrnsd** command.
2. Select disks no longer in use in any file system. Issue the **mmlsnsd -F** command to display the available disks.

The disk may then be used to replace a disk in the file system using the **mmrpldisk** command.

Note:

1. You cannot replace a disk when it is the only remaining disk in the file system.
2. Under no circumstances should you replace a stopped disk. You need to start a stopped disk before replacing it. If a disk cannot be started, you must delete it using the **mmdeldisk** command. See the *General Parallel File System: Problem Determination Guide* and search for *Disk media failure*.
3. The file system need not be unmounted before the **mmrpldisk** command can be run.

Results

Upon successful completion of the **mmrpldisk** command, the disk is replaced in the file system and data is copied to the new disk without restriping.

Parameters

Device

The device name of the file system where the disk is to be replaced. File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

This must be the first parameter.

Disk Name

The name of the disk to be replaced, which was previously passed to the **mmcrfs**, **mmadddisk**, or **mmrpldisk** commands. You can display the entire list of disk names by issuing the **mmlsdisk** command.

DiskDesc

A descriptor for the replacement disk.

-F *DescFile*

Specifies a file containing the disk descriptor for the replacement disk.

The disk descriptor must be specified in the form (second, third, and sixth fields reserved):

DiskName:::DiskUsage:FailureGroup:::

DiskName

You must specify the name of the NSD previously created by the **mmcrnsd** command. For a list of available disks, issue the **mmlsnsd -F** command.

Disk Usage

Specify a disk usage or inherit the disk usage of the disk being replaced:

dataAndMetadata

Indicates that the disk contains both data and metadata.

dataOnly

Indicates that the disk contains data and does not contain metadata.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see *General Parallel File System: Advanced Administration* and search on *Synchronous mirroring utilizing GPFS replication*.

Failure Group

A number identifying the failure group to which this disk belongs. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. If you do not specify a failure group, the new disk inherits the failure group of the disk being replaced.

Note: While it is not absolutely necessary to specify the same disk descriptor parameters for the new disk as the old disk, it is suggested you do so. If the new disk is equivalent in size as the old disk, and if the *DiskUsage* and *FailureGroup* parameters are the same, the data and metadata can be completely migrated from the old disk to the new disk. A disk replacement in this manner allows the file system to maintain its current data and metadata balance.

If the new disk has a different size, *DiskUsage* parameter, or *FailureGroup* parameter, the operation may leave the file system unbalanced and require a restripe. Additionally, a change in size or the *DiskUsage* parameter may cause the operation to fail since other disks in the file system may not have sufficient space to absorb more data or metadata. In this case you must first use the **mmadddisk** command to add the new disk, the **mmdestdisk** command to delete the old disk, and finally the **mmrestripefs** command to rebalance the file system.

-N {Node[,Node...] | NodeFile | NodeClass}

Specify the nodes that participate in the migration of data from the old to the new disk. This command supports all defined node classes. The default is **all** (all nodes in the GPFS cluster will participate in the restripe of the file system).

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

-v {yes | no}

Verify that specified disks do not belong to an existing file system. The default is **-v yes**. Specify **-v no** only when you want to reuse disks that are no longer needed for an existing file system. If the command is interrupted for any reason, you must use the **-v no** option on the next invocation of the command.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmrpldisk** command.

You may issue the **mmrpldisk** command from any node in the GPFS cluster.

When using the **rmp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To replace disk **hd27n01** in **fs1** with a new disk, **hd16vsdn10** allowing the disk usage and failure group parameters to default to the corresponding values of **hd27n01**, and have only nodes **c154n01**, **c154n02**, and **c154n09** participate in the migration of the data, issue this command:

```
mmrpldisk fs1 hd27n01 hd16vsdn10 -N c154n01,c154n02,c154n09
```

The system displays information similar to:

Replacing hd27n01 ...

The following disks of fs1 will be formatted on node c155n01.ppd.pok.ibm.com:

hd16vsdn10: size 17793024 KB

Extending Allocation Map

Checking Allocation Map for storage pool 'system'

```
7 % complete on Wed May 16 16:36:30 2007
18 % complete on Wed May 16 16:36:35 2007
34 % complete on Wed May 16 16:36:40 2007
49 % complete on Wed May 16 16:36:45 2007
65 % complete on Wed May 16 16:36:50 2007
82 % complete on Wed May 16 16:36:55 2007
98 % complete on Wed May 16 16:37:00 2007
100 % complete on Wed May 16 16:37:01 2007
```

Completed adding disks to file system fs1.

Scanning system storage pool

Scanning file system metadata, phase 1 ...

```
2 % complete on Wed May 16 16:37:04 2007
7 % complete on Wed May 16 16:37:11 2007
14 % complete on Wed May 16 16:37:18 2007
20 % complete on Wed May 16 16:37:24 2007
27 % complete on Wed May 16 16:37:31 2007
34 % complete on Wed May 16 16:37:37 2007
50 % complete on Wed May 16 16:37:50 2007
61 % complete on Wed May 16 16:38:00 2007
68 % complete on Wed May 16 16:38:06 2007
79 % complete on Wed May 16 16:38:16 2007
90 % complete on Wed May 16 16:38:26 2007
100 % complete on Wed May 16 16:38:32 2007
```

Scan completed successfully.

Scanning file system metadata, phase 2 ...

Scanning file system metadata for fs1spl storage pool

```
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
  3 % complete on Wed May 16 16:38:38 2007
 25 % complete on Wed May 16 16:38:47 2007
 53 % complete on Wed May 16 16:38:53 2007
 87 % complete on Wed May 16 16:38:59 2007
 97 % complete on Wed May 16 16:39:06 2007
100 % complete on Wed May 16 16:39:07 2007
Scan completed successfully.
Done
mmrpldisk: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

See also

“mmadddisk Command” on page 66

“mmchdisk Command” on page 100

“mmcrnsd Command” on page 137

“mmlsdisk Command” on page 213

“mmlsnsd Command” on page 227

“mmrestripefs Command” on page 268

Location

/usr/lpp/mmfs/bin

mmshutdown Command

Unmounts all GPFS file systems and stops GPFS on one or more nodes.

Synopsis

mmshutdown [-t *UnmountTimeout*] [-a | -N {*Node[,Node...]* | *NodeFile* | *NodeClass*}]

Description

Use the **mmshutdown** command to stop the GPFS daemons on one or more nodes. If no operand is specified, GPFS is stopped only on the node from which the command was issued.

The **mmshutdown** command first attempts to unmount all GPFS file systems. If the unmount does not complete within the specified *timeout* period, the GPFS daemons shut down anyway.

Results

Upon successful completion of the **mmshutdown** command, these tasks are completed:

- GPFS file systems are unmounted.
- GPFS daemons are stopped.

Parameters

-a Stop GPFS on all nodes in a GPFS cluster.

-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}

Directs the **mmshutdown** command to process a set of nodes. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

Options

-t *UnmountTimeout*

The maximum amount of time, in seconds, that the unmount command is given to complete. The default timeout period is equal to:

$60 + 3 \times \text{number of nodes}$

If the unmount does not complete within the specified amount of time, the command times out and the GPFS daemons shut down.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmshutdown** command.

You may issue the **mmshutdown** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To **stop** GPFS on all nodes in the GPFS cluster, issue this command:

```
mmshutdown -a
```

The system displays information similar to:

```
Thu Aug 12 13:10:40 EDT 2004: 6027-1341 mmshutdown: Starting
force unmount of GPFS file systems.
k164n05.kgn.com: forced unmount of /fs1
k164n04.kgn.ibm.com: forced unmount of /fs1
k164n06.kgn.ibm.com: forced unmount of /fs1
Thu Aug 12 13:10:45 EDT 2004: 6027-1344 mmshutdown: Shutting
down GPFS daemons
k164n04.kgn.ibm.com: Shutting down!
k164n06.kgn.ibm.com: Shutting down!
k164n05.kgn.ibm.com: Shutting down!
k164n04.kgn.ibm.com: 'shutdown' command about to kill process
49682
k164n05.kgn.ibm.com: 'shutdown' command about to kill process
28194
k164n06.kgn.ibm.com: 'shutdown' command about to kill process
```

```
Thu Aug 12 13:10:54 EDT 2004: 6027-1345 mmshutdown: Finished
```

2. To stop GPFS on only node **k164n04**, issue this command:

```
mmshutdown -N k164n04
```

The system displays information similar to:

```
mmshutdown -N k164n04
Thu Aug 12 13:12:06 EDT 2004: 6027-1341 mmshutdown: Starting
force unmount of GPFS file systems
k164n04: forced unmount of /fs1
Thu Aug 12 13:12:11 EDT 2004: 6027-1344 mmshutdown: Shutting
down GPFS daemons
k164n04: Shutting down!
k164n04: 'shutdown' command about to kill process 65036
Thu Aug 12 13:12:20 EDT 2004: 6027-1345 mmshutdown: Finished
```

See also

“mmgetstate Command” on page 199

“mmlscluster Command” on page 209

“mmstartup Command” on page 280

Location

/usr/lpp/mmfs/bin

mmsnapdir Command

Creates and deletes invisible directories that connect to the snapshots of a GPFS file system, and changes the name of the snapshots subdirectory

Synopsis

mmsnapdir *Device* **{[-r | -a] [-s SnapDirName]}**

Or,

mmsnapdir *Device* **[-q]**

Description

Use the **mmsnapdir** command to create or delete invisible directories that connect to the snapshots of a GPFS file system, and change the name of the snapshots subdirectory.

Snapshots appear in a subdirectory in the root directory of the file system. If you prefer to access the snapshots from each file system directory rather than traversing through the root directory, you may create an invisible directory to make the connection by issuing the **mmsnapdir** command with the **-a** flag (see Example 1 on page 278). The **-a** flag of the **mmsnapdir** command creates an invisible directory in each normal directory in the active file system (they do not appear in directories in snapshots) that contains a subdirectory for each existing snapshot of the file system. These subdirectories correspond to the copy of the that directory in the snapshot with the same name.

If the **mmsnapdir** command is issued while another snapshot command is running, the **mmsnapdir** command waits for that command to complete.

For more information about GPFS snapshots, see *Creating and maintaining snapshots of GPFS file systems* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system. File system names need not be fully-qualified. **fs0** is just as acceptable as **/dev/fs0**.

This must be the first parameter.

Options

- a** Adds a snapshots subdirectory to all subdirectories in the file system.
- q** Displays current settings, if issued without any other flags.
- r** Reverses the effect of the **-a** option. All invisible snapshot directories are removed. The snapshot directory under the file system root directory is not affected.
- s SnapDirName**
Changes the name of the snapshots subdirectory to *SnapDirName*. This affects both the directory in the file system root as well as the invisible directory in the other file system directories if the **mmsnapdir -a** command has been issued.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

If you are a root user, you may issue the **mmsnapdir** command from any node in the GPFS cluster.

You must be a root user to use the **-a**, **-r**, and **-s** options.

If you are a non-root user, you may only specify file systems that belong to the same cluster as the node on which the **mmsnapdir** command was issued.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. To rename the **.snapshots** directory (the default snapshots directory name) to **.link** for file system **fs1**, issue the command:

```
mmsnapdir fs1 -s .link
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.link/snap1/file1
/fs1/.link/snap1/userA/file2
/fs1/.link/snap1/userA/file3
```

2. Issuing:

```
mmsnapdir fs1 -a
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/userA/.link/snap1/file2
/fs1/userA/.link/snap1/file3

/fs1/.link/snap1/file1
/fs1/.link/snap1/userA/file2
/fs1/.link/snap1/userA/file3
```

The **.link** subdirectory under the root directory and under each subdirectory of the tree provides two different paths to each snapshot copy of a file. For example, **/fs1/userA/.link/snap1/file2** and **/fs1/.link/snap1/userA/file2** are two different paths that access the same snapshot copy of **/fs1/userA/file2**.

3. Issuing:

```
mmsnapdir fs1 -r
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.link/snap1/file1
/fs1/.link/snap1/userA/file2
/fs1/.link/snap1/userA/file3
```


4. Issuing:

```
mmsnapdir fs1 -q
```

The output is similar to this:

Snapshot directory for "fs1" is ".link" (root directory only)

See also

“mmcrsnapshot Command” on page 142

“mmdelsnapshot Command” on page 174

“mmlssnapshot Command” on page 235

“mmrestorefs Command” on page 262

Location

/usr/lpp/mmfs/bin

mmstartup Command

Starts the GPFS subsystem on one or more nodes.

Synopsis

mmstartup [-a | -N {*Node[,Node...]* | *NodeFile* | *NodeClass*}] [-E *EnvVar=value ...*]

Description

Use the **mmstartup** command to start the GPFS daemons on one or more nodes. If no operand is specified, GPFS is started only on the node from which the command was issued.

Parameters

-a Start GPFS on all nodes in a GPFS cluster.

-N {*Node[,Node...]* | *NodeFile* | *NodeClass*}

Directs the **mmstartup** command to process a set of nodes. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

Options

-E *EnvVar=value*

Specifies the name and value of an environment variable to be passed to the GPFS daemon. You can specify multiple **-E** options.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmstartup** command.

You may issue the **mmstartup** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

To start GPFS on all nodes in the GPFS cluster, issue this command:

```
mmstartup -a
```

The system displays information similar to:

```
Thu Aug 12 13:22:40 EDT 2004: 6027-1642 mmstartup: Starting GPFS ...
```

See also

“mmgetstate Command” on page 199

“mmiscluster Command” on page 209

“mmshutdown Command” on page 275

Location

/usr/lpp/mmfs/bin

mmtracectl Command

Sets up and enables GPFS tracing.

Synopsis

```
mmtracectl {--start | --stop | --off | --set} [--trace={io | all | def | "Class Level [Class Level ...]"}]
[--trace-recycle={off | local | global | globalOnShutdown}] [--aix-trace-buffer-size=BufferSize]
[--tracedev-buffer-size=BufferSize] [--trace-file-size=FileSize] [--trace-dispatch={yes | no}] [-N {Node
[,Node...]} | NodeFile | NodeClass]
```

Description

Attention: Use this command only under the direction of your IBM service representative.

Use the **mmtracectl** command to perform the following functions:

- Start or stop tracing.
- Turn tracing on (start or set trace recycle) or off on the next session. This is a persistent setting to automatically start trace each time GPFS starts).
- Allow for predefined trace levels: **io**, **all**, and **def**, as well as user-specified trace levels.
- Allow for changing the size of trace buffer sizes for AIX and all others using the **tracedev** option.
- Trace recycle functions, which allow for never cycling traces (**off** option), tracing on all nodes when GPFS ends abnormally (**global** option), cycling traces any time GPFS goes down on all nodes (**globalonshutdown** option).

Results

GPFS tracing can be started, stopped, or related configuration options can be set.

Parameters

- | **--start | --stop | --off | --set**
Specifies the actions that the **mmtracectl** command performs, where:
 - start** Starts the trace.
 - stop** Stops the trace.
 - off** Clears all of the setting variables and stops the trace.
 - set** Sets the trace variables.
- trace={io | all | def | "Class Level [Class Level ...]"}**
Allows for predefined and user-specified trace levels, where:
 - io** Indicates trace-level settings tailored for input and output (I/O).
 - all** Sets trace levels to their highest setting (9).
 - def** Indicates that the default trace settings will be used.**"Class Level [Class Level ...]"**
Specifies a trace class level.
- traceRecycle={off | local | global | globalOnShutdown}**
Controls trace recycling during daemon termination. The following values are recognized:
 - off** Does not recycle traces. This is the default.
 - local** Recycles traces on the local node every time **mmfsd** goes down.
 - global** Recycles traces on all nodes in the cluster when an abnormal daemon shutdown occurs.

globalOnShutdown

Recycles traces on all nodes in the cluster for normal and abnormal daemon shutdowns.

-N {Node[,Node...] | NodeFile | NodeClass }

Specifies the nodes that will participate in the tracing of the file system. This option supports all defined node classes. The default value is **all**.

For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

Options

--aix-trace-buffer-size =BufferSize

Controls the size of the trace buffer in memory for AIX.

--tracedev-buffer-size=BufferSize

Controls the size of the trace buffer in memory.

--trace-file-Size=FileSize

Controls the size of the trace file.

--trace-dispatch

Enables AIX thread dispatching trace hooks.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmtracectl** command.

You can issue the **mmtracectl** command from any node in the GPFS cluster.

Examples

To set trace levels to the defined group of **def** and have traces start on all nodes when GPFS comes up, issue this command:

```
mmtracectl --set --trace=def --trace-recycle=global
```

The system displays output similar to:

```
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

To confirm the change, issue this command:

```
mmisconfig|grep trace
```

The system displays output similar to:

```
trace all 4 tm 2 thread 1 mutex 1 vnode 2 ksvfs 3 klockl 2 io 3 pgallo 1 mb 1 lock 2 fsck 3
traceRecycle global
```

To manually start traces on all nodes, issue this command:

```
mmtracectl --start
```

See also

“mmchconfig Command” on page 92

mmtrace shell script

Location

/usr/lpp/mmfs/bin

mmumount Command

Unmounts GPFS file systems on one or more nodes in the cluster.

Synopsis

| **mmumount** {*Device* | *MountPoint* | *DriveLetter* | **all** | **all_local** | **all_remote**} [-f] [-a | -N {*Node*[,*Node*...] | *NodeFile* | *NodeClass*}]

Or,

mmumount *Device* -f -C {**all_remote** | *ClusterName*} [-N *Node*[,*Node*...]]

Description

Another name for the **mmumount** command is the **mmunmount** command. Either name can be used.

The **mmumount** command unmounts a previously mounted GPFS file system on one or more nodes in the cluster. If no nodes are specified, the file systems are unmounted only on the node from which the command was issued. The file system can be specified using its device name or the mount point where it is currently mounted.

Use the first form of the command to unmount file systems on nodes that belong to the local cluster.

Use the second form of the command with the **-C** option when it is necessary to force an unmount of file systems that are owned by the local cluster, but are mounted on nodes that belong to another cluster.

When a file system is unmounted by force with the second form of the **mmumount** command, the affected nodes may still show the file system as mounted, but the data will not be accessible. It is the responsibility of the system administrator to clear the mount state by issuing the **umount** command.

| When multiple nodes are affected and the unmount target is identified via a mount point or a Windows
| drive letter, the mount point is resolved on each of the target nodes. Depending on how the file systems
| were mounted, this may result in different file systems being unmounted on different nodes. When in
| doubt, always identify the target file system with its device name.

Parameters

| *Device* | *MountPoint* | *DriveLetter* | **all** | **all_local** | **all_remote**

Indicates the file system or file systems to be unmounted.

Device

Is the device name of the file system to be unmounted. File system names do not need to be fully qualified. **fs0** is as acceptable as **/dev/fs0**.

MountPoint

Is the location where the GPFS file system to be unmounted is currently mounted.

| *DriveLetter*

| Identifies a file system by its Windows drive letter.

all Indicates all file systems that are known to this cluster.

all_local
Indicates all file systems that are owned by this cluster.

all_remote
Indicates all files systems that are owned by another cluster to which this cluster has access.

This must be the first parameter.

Options

- a** Unmounts the file system on all nodes in the GPFS cluster.
- f** Forces the unmount to take place even though the file system may be still in use.
Use this flag with *extreme caution*. Using this flag may cause outstanding write operations to be lost. Because of this, forcing an unmount can cause data integrity failures and should be used with caution.
- C {all_remote | ClusterName}**
Specifies the cluster on which the file system is to be unmounted by force. **all_remote** denotes all clusters other than the one from which the command was issued.
- N {Node[,Node...] | NodeFile | NodeClass}**
Specifies the nodes on which the file system is to be unmounted. For general information on how to specify node names, see “Specifying nodes as input to GPFS commands” on page 2.

This command does not support a *NodeClass* of **mount**.

When the **-N** option is specified in conjunction with **-C ClusterName**, the specified node names are assumed to refer to nodes that belong to the specified remote cluster (as identified by the **mmlsmount** command). The **mmumount** command cannot verify the accuracy of this information. *NodeClass* and *NodeFile* are not supported in conjunction with the **-C** option.

Exit status

- 0** Successful completion.
- nonzero**
A failure has occurred.

Security

You must have root authority to run the **mmumount** command.

You may issue the **mmumount** command from any node in the GPFS cluster.

Examples

1. To unmount file system **fs1** on all nodes in the cluster, issue this command:

```
mmumount fs1 -a
```

The system displays output similar to:

```
Fri Feb 10 15:51:25 EST 2006: mmumount: Unmounting file systems ...
```

2. To force unmount file system **fs2** on the local node, issue this command:

```
mmumount fs2 -f
```

The system displays output similar to:

```
Fri Feb 10 15:52:20 EST 2006: mmumount: Unmounting file systems ...  
forced unmount of /fs2
```

See also

“mmmount Command” on page 237

“mmlsmount Command” on page 225

Location

`/usr/lpp/mmfs/bin`

mmunlinkfileset Command

Removes the junction to the a GPFS fileset.

Synopsis

mmunlinkfileset *Device* {*FilesetName* | **-J** *JunctionPath*} [**-f**]

Description

The **mmunlinkfileset** command removes the junction to the fileset. The junction can be specified by path or by naming the fileset that is its target. The unlink fails if there are files open in the fileset, unless the **-f** flag is specified. The root fileset may not be unlinked.

Attention: If you are using the TSM Backup Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

For information on GPFS filesets, see the chapter *Policy-based data management for GPFS* in *General Parallel File System: Advanced Administration Guide*.

Parameters

Device

The device name of the file system that contains the fileset.

File system names need not be fully-qualified. **fs0** is as acceptable as **/dev/fs0**.

FilesetName

Specifies the name of the fileset to be removed.

-J *JunctionPath*

Specifies the name of the junction to be removed.

A junction is a special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

Options

-f Forces the unlink to take place even though there may be open files. This option forcibly closes any open files, causing an **errno** of **ESTALE** on their next use of the file.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmunlinkfileset** command.

You may issue the **mmunlinkfileset** command from any node in the GPFS cluster.

When using the **rcp** and **rsh** commands for remote communication, a properly configured **.rhosts** file must exist in the root user's home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the **mmcrcluster** or the **mmchcluster** command, you must ensure:

1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password, and without any extraneous messages.

Examples

1. This command indicates the current configuration of filesets for file system **gpfs1**:

```
mmllsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
root      Linked   /gpfs1
fset1     Linked   /gpfs1/fset
```

This command unlinks fileset **fset1** from file system **gpfs1**:

```
mmunlinkfileset gpfs1 fset1
```

The system displays output similar to:

```
Fileset 'fset1' unlinked.
```

To confirm the change, issue this command:

```
mmllsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
root      Linked   /gpfs1
fset1     Unlinked --
```

2. This command indicates the current configuration of filesets for file system **gpfs1**:

```
mmllsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
root      Linked   /gpfs1
fset1     Linked   /gpfs1/fset1
```

This command unlinks junction path **/gpfs1/fset1** from file system **gpfs1**:

```
mmunlinkfileset gpfs1 -J /gpfs1/fset1
```

The system displays output similar to:

```
Fileset 'fset1' unlinked.
```

To confirm the change, issue this command:

```
mmllsfileset gpfs1
```

The system displays output similar to:

```
Filesets in file system 'gpfs1':
Name      Status   Path
root      Linked   /gpfs1
fset1     Unlinked --
```

See also

“mmchfileset Command” on page 107

“mmcrfileset Command” on page 128

“mmdelfileset Command” on page 165

“mmlinkfileset Command” on page 205

“mmisfileset Command” on page 216

Location

/usr/lpp/mmfs/bin

Chapter 9. GPFS programming interfaces

A list of all the GPFS programming interfaces and a short description of each is presented in this topic.

- I The GPFS APIs are not supported on Windows.

Table 8 summarizes the GPFS programming interfaces.

Table 8. GPFS programming interfaces

Interface	Purpose
"gpfs_acl_t Structure" on page 293	Contains buffer mapping for the gpfs_getacl() and gpfs_putacl() subroutines.
"gpfs_close_inodescan() Subroutine" on page 294	Closes an inode scan.
"gpfs_cmp_fssnapid() Subroutine" on page 295	Compares two file system snapshot IDs.
"gpfs_direntx_t Structure" on page 297	Contains attributes of a GPFS directory entry.
"gpfs_fcntl() Subroutine" on page 298	Performs operations on an open file.
"gpfs_fgetattrs() Subroutine" on page 301	Retrieves all extended file attributes in opaque format.
"gpfs_fputattrs() Subroutine" on page 303	Sets all of the extended file attributes for a file.
"gpfs_fputattrswithpathname() Subroutine" on page 305	Sets all of the extended file attributes for a file and invokes the policy engine for RESTORE rules.
"gpfs_free_fssnaphandle() Subroutine" on page 307	Frees a file system snapshot handle.
"gpfs_fssnap_handle_t Structure" on page 308	Contains a handle for a GPFS file system or snapshot.
"gpfs_fssnap_id_t Structure" on page 309	Contains a permanent identifier for a GPFS file system or snapshot.
"gpfs_fstat() Subroutine" on page 310	Returns exact file status for a GPFS file.
"gpfs_get_fsname_from_fssnaphandle() Subroutine" on page 312	Obtains a file system name from its snapshot handle.
"gpfs_get_fssnaphandle_by_fssnapid() Subroutine" on page 313	Obtains a file system snapshot handle using its snapshot ID.
"gpfs_get_fssnaphandle_by_name() Subroutine" on page 314	Obtains a file system snapshot handle using its name.
"gpfs_get_fssnaphandle_by_path() Subroutine" on page 316	Obtains a file system snapshot handle using its path name.
"gpfs_get_fssnapid_from_fssnaphandle() Subroutine" on page 318	Obtains a file system snapshot ID using its snapshot handle.
"gpfs_get_pathname_from_fssnaphandle() Subroutine" on page 320	Obtains a file system path name using its snapshot handle.
"gpfs_get_snapdirname() Subroutine" on page 321	Obtains the name of the directory containing snapshots.
"gpfs_get_snapname_from_fssnaphandle() Subroutine" on page 323	Obtains a snapshot name using its file system snapshot handle.
"gpfs_getacl() Subroutine" on page 324	Retrieves the access control information for a GPFS file.
"gpfs_iattr_t Structure" on page 326	Contains attributes of a GPFS inode.
"gpfs_icolse() Subroutine" on page 329	Closes a file given its inode file handle.
"gpfs_ifile_t Structure" on page 330	Contains a handle for a GPFS inode.
"gpfs_igetattrs() Subroutine" on page 331	Obtains extended file attributes.
"gpfs_igetfilessetName() Subroutine" on page 333	Returns the name of the fileset defined by a fileset ID.

Table 8. GPFS programming interfaces (continued)

Interface	Purpose
"gpfs_igetstoragepool() Subroutine" on page 335	Returns the name of the storage pool for the given storage pool ID.
"gpfs_iopen() Subroutine" on page 337	Opens a file or directory by its inode number.
"gpfs_iread() Subroutine" on page 339	Reads a file opened by gpfs_iopen() .
"gpfs_ireaddir() Subroutine" on page 341	Reads the next directory entry.
"gpfs_ireadlink() Subroutine" on page 343	Reads a symbolic link.
"gpfs_ireadx() Subroutine" on page 345	Performs block level incremental read of a file within an incremental inode scan.
"gpfs_iscan_t Structure" on page 347	Contains mapping of the inode scan structure.
"gpfs_next_inode() Subroutine" on page 348	Retrieves the next inode from the inode scan.
"gpfs_opaque_acl_t Structure" on page 350	Contains buffer mapping for the gpfs_getacl() and gpfs_putacl() subroutines.
"gpfs_open_inodescan() Subroutine" on page 351	Opens an inode scan over a file system or snapshot.
"gpfs_prealloc() Subroutine" on page 353	Pre-allocates disk storage for a GPFS file.
"gpfs_putacl() Subroutine" on page 355	Sets the access control information for a GPFS file.
"gpfs_quotactl() Subroutine" on page 357	Manipulates disk quotas on file systems.
"gpfs_quotaInfo_t Structure" on page 360	Contains buffer mapping for the gpfs_quotactl() subroutine.
"gpfs_seek_inode() Subroutine" on page 362	Advances an inode scan to the specified inode number.
"gpfs_stat() Subroutine" on page 364	Returns exact file status for a GPFS file.
"gpfsAccessRange_t Structure" on page 366	Declares an access range within a file for an application.
"gpfsCancelHints_t Structure" on page 367	Indicates to remove any hints against the open file handle.
"gpfsClearFileCache_t Structure" on page 368	Indicates file access in the near future is not expected.
"gpfsDataShipMap_t Structure" on page 369	Indicates which agent nodes are to be used for data shipping.
"gpfsDataShipStart_t Structure" on page 371	Initiates data shipping mode.
"gpfsDataShipStop_t Structure" on page 374	Takes a file out of data shipping mode.
"gpfsFcntlHeader_t Structure" on page 375	Contains declaration information for the gpfs_fcntl() subroutine.
"gpfsFreeRange_t Structure" on page 376	Undeclares an access range within a file for an application.
"gpfsGetFilesetName_t Structure" on page 377	Obtains a file's fileset name.
"gpfsGetReplication_t Structure" on page 378	Obtains a file's replication factors.
"gpfsGetSnapshotName_t Structure" on page 380	Obtains a file's snapshot name.
"gpfsGetStoragePool_t Structure" on page 381	Obtains a file's storage pool name.
"gpfsMultipleAccessRange_t Structure" on page 382	Defines prefetching and write-behind file access for an application.
"gpfsRestripeData_t Structure" on page 384	Restripes a file's data blocks.
"gpfsSetReplication_t Structure" on page 385	Sets a file's replication factors.
"gpfsSetStoragePool_t Structure" on page 387	Sets a file's assigned storage pool.

gpfs_acl_t Structure

Contains buffer mapping for the **gpfs_getacl()** and **gpfs_putacl()** subroutines.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
/* The GPFS ACL */
typedef struct gpfs_acl
{
    gpfs_aclLen_t  acl_len; /* Total length of this ACL in bytes */
    gpfs_aclLevel_t acl_level; /* Reserved (must be zero) */
    gpfs_aclVersion_t acl_version; /* POSIX or NFS4 ACL */
    gpfs_aclType_t  acl_type; /* Access, Default, or NFS4 */
    gpfs_aclCount_t acl_nace; /* Number of Entries that follow */
    union
    {
        gpfs_ace_v1_t ace_v1[1]; /* when GPFS_ACL_VERSION_POSIX */
        gpfs_ace_v4_t ace_v4[1]; /* when GPFS_ACL_VERSION_NFS4 */
    };
} gpfs_acl_t;
```

Description

The **gpfs_acl_t** structure contains size, version, and ACL type information for the **gpfs_getacl()** and **gpfs_putacl()** subroutines.

Members

acl_len

The total length (in bytes) of this **gpfs_acl_t** structure.

acl_level

Reserved for future use. Currently must be zero.

acl_version

This field contains the version of the GPFS ACL. GPFS supports two ACL versions: **GPFS_ACL_VERSION_POSIX** and **GPFS_ACL_VERSION_NFS4**. On input to the **gpfs_getacl()** subroutine, set this field to zero.

acl_type

On input to the **gpfs_getacl()** subroutine, set this field to:

- Either **GPFS_ACL_TYPE_ACCESS** or **GPFS_ACL_TYPE_DEFAULT** for POSIX ACLs
- **GPFS_ACL_TYPE_NFS4** for NFS ACLs.

These constants are defined in the **gpfs.h** header file.

acl_nace

The number of ACL entries that are in the array (**ace_v1** or **ace_v4**).

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_close_inodescan() Subroutine

Closes an inode scan.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
void gpfs_close_inodescan(gpfs_iscan_t *iscan);
```

Description

The **gpfs_close_inodescan()** subroutine closes the scan of the inodes in a file system or snapshot that was opened with the **gpfs_open_inodescan()** subroutine. The **gpfs_close_inodescan()** subroutine frees all storage used for the inode scan and invalidates the **iscan** handle.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

iscan Pointer to the inode scan handle.

Exit status

The **gpfs_close_inodescan()** subroutine returns void.

Exceptions

None.

Error status

None.

Examples

For an example using **gpfs_close_inodescan()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_cmp_fssnapid() Subroutine

Compares two snapshot IDs for the same file system to determine the order in which the two snapshots were taken.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_cmp_fssnapid (const gpfs_fssnap_id_t *fssnapId1,
                      const gpfs_fssnap_id_t *fssnapId2,
                      int *result);
```

Description

The **gpfs_cmp_fssnapid()** subroutine compares two snapshot IDs for the same file system to determine the order in which the two snapshots were taken. The **result** parameter is set as follows:

- **result** less than zero indicates that snapshot 1 was taken before snapshot 2.
- **result** equal to zero indicates that snapshot 1 and 2 are the same.
- **result** greater than zero indicates that snapshot 1 was taken after snapshot 2.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fssnapId1

File system snapshot ID of the first snapshot.

fssnapId2

File system snapshot ID of the second snapshot.

result Pointer to an integer indicating the outcome of the comparison.

Exit status

If the **gpfs_cmp_fssnapid()** subroutine is successful, it returns a value of 0 and the **result** parameter is set as described above.

If the **gpfs_cmp_fssnapid()** subroutine is unsuccessful, it returns a value of -1 and the global error variable **errno** is set to indicate the nature of the error.

Exceptions

None.

Error status

EDOM The two snapshots cannot be compared because they were taken from two different file systems.

ENOSYS

The **gpfs_cmp_fssnapid()** subroutine is not available.

GPFS_E_INVALID_FSSNAPHANDLE

The file system snapshot handle is not valid.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_direntx_t Structure

Contains attributes of a GPFS directory entry.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_direntx
{
    int            d_version; /*this struct's version*/
    unsigned short d_reclen; /*actual size of this struct including
                               null-terminated variable-length d_name*/
    unsigned short d_type;    /*types are defined below*/
    gpfs_ino_t     d_ino;     /*file inode number*/
    gpfs_gen_t     d_gen;     /*generation number for the inode*/
    char           d_name[256]; /*null-terminated variable-length name*/
} gpfs_direntx_t;

/* File types for d_type field in gpfs_direntx_t */
#define GPFS_DE_OTHER    0
#define GPFS_DE_DIR      4
#define GPFS_DE_REG      8
#define GPFS_DE_LNK     10
```

Description

The **gpfs_direntx_t** structure contains the attributes of a GPFS directory entry.

Members

d_version

The version number of this structure.

d_reclen

The actual size of this structure including the null-terminated variable-length **d_name** field.

To allow some degree of forward compatibility, careful callers should use the **d_reclen** field for the size of the structure rather than the **sizeof()** function.

d_type

The type of directory.

d_ino The directory inode number.

d_gen The directory generation number.

d_name

Null-terminated variable-length name of the directory.

Examples

For an example using **gpfs_direntx_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fcntl() Subroutine

Performs operations on an open file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fcntl(int fileDesc, void* fcntlArgP)
```

Description

The **gpfs_fcntl()** subroutine is used to pass file access pattern information and to control certain file attributes on behalf of an open file. More than one operation can be requested with a single invocation of **gpfs_fcntl()**. The type and number of operations is determined by the second operand, **fcntlArgP**, which is a pointer to a data structure built in memory by the application. This data structure consists of:

- A fixed length header, mapped by **gpfsFcntlHeader_t**.
- A variable list of individual file access hints, directives or other control structures:
 - File access hints:
 1. **gpfsAccessRange_t**
 2. **gpfsFreeRange_t**
 3. **gpfsMultipleAccessRange_t**
 4. **gpfsClearFileCache_t**
 - File access directives:
 1. **gpfsCancelHints_t**
 2. **gpfsDataShipMap_t**
 3. **gpfsDataShipStart_t**
 4. **gpfsDataShipStop_t**
 - Other file attribute operations:
 1. **gpfsGetFilesetName_t**
 2. **gpfsGetReplication_t**
 3. **gpfsGetSnapshotName_t**
 4. **gpfsGetStoragePool_t**
 5. **gpfsRestripeData_t**
 6. **gpfsSetReplication_t**
 7. **gpfsSetStoragePool_t**

The above hints, directives and other operations may be mixed within a single **gpfs_fcntl()** subroutine, and are performed in the order that they appear. A subsequent hint or directive may cancel out a preceding one. See Chapter 7, “Communicating file access patterns to GPFS,” on page 61.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fileDesc

The file descriptor identifying the file to which GPFS applies the hints and directives.

fcntlArgP

A pointer to the list of operations to be passed to GPFS.

Exit status

If the **gpfs_fcntl()** subroutine is successful, it returns a value of 0.

If the **gpfs_fcntl()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EBADF

The file descriptor is not valid.

EINVAL

The file descriptor does not refer to a GPFS file or a regular file.

The system call is not valid.

ENOSYS

The **gpfs_fcntl()** subroutine is not supported under the current file system format.

Examples

1. This programming segment releases all cache data held by the file *handle* and tell GPFS that the subroutine will write the portion of the file with file offsets between 2 GB and 3 GB minus one:

```
struct
{
    gpfsFcntlHeader_t hdr;
    gpfsClearFileCache_t rel;
    gpfsAccessRange_t acc;
} arg;

arg.hdr.totalLength = sizeof(arg);
arg.hdr.fcntlVersion = GPFS_FCNTL_CURRENT_VERSION;
arg.hdr.fcntlReserved = 0;
arg.rel.structLen = sizeof(arg.rel);
arg.rel.structType = GPFS_CLEAR_FILE_CACHE;
arg.acc.structLen = sizeof(arg.acc);
arg.acc.structType = GPFS_ACCESS_RANGE;
arg.acc.start = 2LL * 1024LL * 1024LL * 1024LL;
arg.acc.length = 1024 * 1024 * 1024;
arg.acc.isWrite = 1;
rc = gpfs_fcntl(handle, &arg);
```

2. This programming segment gets the storage pool name and fileset name of a file from GPFS.

```
struct {
    gpfsFcntlHeader_t hdr;
    gpfsGetStoragePool_t pool;
    gpfsGetFilesetName_t fileset;
} fcntlArg;

fcntlArg.hdr.totalLength = sizeof(fcntlArg.hdr) + sizeof(fcntlArg.pool) + sizeof(fcntlArg.fileset);
fcntlArg.hdr.fcntlVersion = GPFS_FCNTL_CURRENT_VERSION;
fcntlArg.hdr.fcntlReserved = 0;
```

```
fcntlArg.pool.structLen = sizeof(fcntlArg.pool);
fcntlArg.pool.structType = GPFS_FCNTL_GET_STORAGEPOOL;

fcntlArg.fileset.structLen = sizeof(fcntlArg.fileset);
fcntlArg.fileset.structType = GPFS_FCNTL_GET_FILESETNAME;

rc = gpfs_fcntl(fd, &fcntlArg);
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fgetattrs() Subroutine

Retrieves all extended file attributes in opaque format.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fgetattrs(int fileDesc, int flags, void *bufferP,
                  int bufferSize, int *attrSizeP)
```

Description

The **gpfs_fgetattrs()** subroutine, together with **gpfs_fputattrs()**, is intended for use by a backup program to save (**gpfs_fgetattrs()**) and restore (**gpfs_fputattrs()**) extended file attributes such as ACLs, DMAPI attributes, and other information for the file. If the file has no extended attributes, the **gpfs_fgetattrs()** subroutine returns a value of 0, but sets **attrSizeP** to 0 and leaves the contents of the buffer unchanged.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fileDesc

The file descriptor identifying the file whose extended attributes are being retrieved.

flags Must have one of these three values:

GPFS_ATTRFLAG_DEFAULT

- Saves the attributes for file placement
- Saves the currently assigned storage pool

GPFS_ATTRFLAG_NO_PLACEMENT

- Does not save attributes for file placement
- Does not save the currently assigned storage pool

GPFS_ATTRFLAG_IGNORE_POOL

- Saves attributes for file placement
- Does not save the currently assigned storage pool

bufferP

Pointer to a buffer to store the extended attribute information.

bufferSize

The size of the buffer that was passed in.

attrSizeP

If successful, returns the actual size of the attribute information that was stored in the buffer. If the *bufferSize* was too small, returns the minimum buffer size.

Exit status

If the **gpfs_fgetattrs()** subroutine is successful, it returns a value of 0.

If the **gpfs_fgetattrs()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

| EINVAL

The file descriptor does not refer to a GPFS file or a regular file.

ENOSPC

bufferSize is too small to return all of the attributes. On return, **attrSizeP* is set to the required size.

ENOSYS

The **gpfs_fgetattrs()** subroutine is not supported under the current file system format.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fputattrs() Subroutine

Sets all the extended file attributes for a file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fputattrs(int fileDesc, int flags, void *bufferP)
```

Description

The **gpfs_fputattrs()** subroutine, together with **gpfs_fgetattrs()**, is intended for use by a backup program to save (**gpfs_fgetattrs()**) and restore (**gpfs_fputattrs()**) all of the extended attributes of a file. This subroutine also sets the storage pool for the file and sets data replication to the values that are saved in the extended attributes.

If the saved storage pool is not valid or if the *IGNORE_POOL* flag is set, GPFS will select the storage pool by matching a **PLACEMENT** rule using the saved file attributes. If GPFS fails to match a placement rule or if there are no placement rules installed, GPFS assigns the file to the system storage pool.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fileDesc

The file descriptor identifying the file whose extended attributes are being set.

flags Non-placement attributes such as ACLs are always restored, regardless of value of the flag. Flags must have one of these three values:

GPFS_ATTRFLAG_DEFAULT

Restores the previously assigned storage pool and previously assigned data replication.

GPFS_ATTRFLAG_NO_PLACEMENT

Does not change storage pool and data replication.

GPFS_ATTRFLAG_IGNORE_POOL

Selects storage pool and data replication by matching the saved attributes to a placement rule instead of restoring the saved storage pool.

bufferP

A pointer to the buffer containing the extended attributes for the file.

If you specify a value of NULL, all extended ACLs for the file are deleted.

Exit status

If the **gpfs_fputattrs()** subroutine is successful, it returns a value of 0.

If the **gpfs_fputattrs()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EINVAL

The buffer pointed to by *bufferP* does not contain valid attribute data.

ENOSYS

The **gpfs_fputattrs()** subroutine is not supported under the current file system format.

Examples

To copy extended file attributes from file **f1** to file **f2**:

```
char buf[4096];
int f1, f2, attrSize, rc;

rc = gpfs_fgetattrs(f1, GPFS_ATTRFLAG_DEFAULT, buf, sizeof(buf), &attrSize);
if (rc != 0)
    ... // error handling
if (attrSize != 0)
    rc = gpfs_fputattrs(f2, 0, buf); //copy attributes from f1 to f2
else
    rc = gpfs_fputattrs(f2, 0, NULL); // f1 has no attributes
                                        // delete attributes on f2
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fputattrswithpathname() Subroutine

Sets all of the extended file attributes for a file and invokes the policy engine for **RESTORE** rules.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fputattrswithpathname(int fileDesc, int flags, void *bufferP, const char *pathname)
```

Description

The **gpfs_fputattrswithpathname()** subroutine sets all of the extended attributes of a file. In addition, **gpfs_fputattrswithpathname()** invokes the policy engine using the saved attributes to match a **RESTORE** rule to set the storage pool and the data replication for the file. The caller should include the full path to the file (including the file name) to allow rule selection based on file name or path. If the file fails to match a **RESTORE** rule or if there are no **RESTORE** rules installed, GPFS selects the storage pool and data replication as it does when calling **gpfs_fputattrs()**.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from one of the following libraries:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fileDesc

Is the file descriptor that identifies the file whose extended attributes are to be set.

flags Non-placement attributes such as ACLs are always restored, regardless of value of the flag. Flags must have one of these three values:

GPFS_ATTRFLAG_DEFAULT

Uses the saved attributes to match a **RESTORE** rule to set the storage pool and the data replication for the file.

GPFS_ATTRFLAG_NO_PLACEMENT

Does not change storage pool and data replication.

GPFS_ATTRFLAG_IGNORE_POOL

Checks the file to see if it matches a **RESTORE** rule. If the file fails to match a **RESTORE** rule, GPFS ignores the saved storage pool and selects a pool by matching the saved attributes to a **PLACEMENT** rule.

bufferP

A pointer to the buffer containing the extended attributes for the file.

If you specify a value of **NULL**, all extended ACLs for the file are deleted.

Exit status

If the **gpfs_fputattrswithpathname()** subroutine is successful, it returns a value of 0.

If the **gpfs_fputattrswithpathname()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EINVAL

The buffer to which *bufferP* points does not contain valid attribute data.

| ENOENT

| No such file or directory.

ENOSYS

The **gpfs_fputattrswithpathname()** subroutine is not supported under the current file system format.

Examples

Refer to “gpfs_fputattrs() Subroutine” on page 303 for examples.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_free_fssnaphandle() Subroutine

Frees a GPFS file system snapshot handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
void gpfs_free_fssnaphandle(gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The **gpfs_free_fssnaphandle()** subroutine frees the snapshot handle that is passed. The return value is always void.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

Exit status

The **gpfs_free_fssnaphandle()** subroutine always returns void.

Exceptions

None.

Error status

None.

Examples

For an example using **gpfs_free_fssnaphandle()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fssnap_handle_t Structure

Contains a handle for a GPFS file system or snapshot.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_fssnap_handle gpfs_fssnap_handle_t;
```

Description

A file system or snapshot is uniquely identified by an **fssnapId** of type **gpfs_fssnap_id_t**. While the **fssnapId** is permanent and global, a shorter **fssnapHandle** is used by the backup application programming interface to identify the file system and snapshot being accessed. The **fssnapHandle**, like a POSIX file descriptor, is volatile and may be used only by the program that created it.

There are three ways to create a file system snapshot handle:

1. By using the name of the file system and snapshot
2. By specifying the path through the mount point
3. By providing an existing file system snapshot ID

Additional subroutines are provided to obtain the permanent, global **fssnapId** from the **fssnapHandle**, or to obtain the path or the names for the file system and snapshot, if they are still available in the file system.

The file system must be mounted in order to use the backup programming application interface. If the **fssnapHandle** is created by the path name, the path may be relative and may specify any file or directory in the file system. Operations on a particular snapshot are indicated with a path to a file or directory within that snapshot. If the **fssnapHandle** is created by name, the file system's unique name may be specified (for example, **fs1**) or its device name may be provided (for example, **/dev/fs1**). To specify an operation on the active file system, the pointer to the snapshot's name should be set to NULL or a zero-length string provided.

The name of the directory under which all snapshots appear may be obtained by the **gpfs_get_snapdirname()** subroutine. By default this is **.snapshots**, but it can be changed using the **mmsnapdir** command. The **gpfs_get_snapdirname()** subroutine returns the currently set value, which is the one that was last set by the **mmsnapdir** command, or the default, if it was never changed.

Members

gpfs_fssnap_handle

File system snapshot handle

Examples

For an example using **gpfs_fssnap_handle_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fssnap_id_t Structure

Contains a permanent, globally unique identifier for a GPFS file system or snapshot.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_fssnap_id
{
    char opaque[48];
} gpfs_fssnap_id_t;
```

Description

A file system or snapshot is uniquely identified by an **fssnapId** of type **gpfs_fssnap_id_t**. The **fssnapId** is a permanent and global identifier that uniquely identifies an active file system or a read-only snapshot of a file system. Every snapshot of a file system has a unique identifier that is also different from the identifier of the active file system itself.

The **fssnapId** is obtained from an open **fssnapHandle**. Once obtained, the **fssnapId** should be stored along with the file system's data for each backup. The **fssnapId** is required to generate an incremental backup. The **fssnapId** identifies the previously backed up file system or snapshot and allows the inode scan to return only the files and data that have changed since that previous scan.

Members

opaque

A 48 byte area for containing the snapshot identifier.

Examples

For an example using **gpfs_fssnap_id_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_fstat() Subroutine

Returns exact file status for a GPFS file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_fstat(int fileDesc, struct stat64 *Buffer)
```

Description

The **gpfs_fstat()** subroutine is used to obtain exact information about the file associated with the *FileDesc* parameter. This subroutine is provided as an alternative to the **stat()** subroutine, which may not provide exact **mtime** and **atime** values. See “Exceptions to Open Group technical standards” on page 393.

read, **write**, or **execute** permission for the named file is not required, but all directories listed in the path leading to the file must be searchable. The file information is written to the area specified by the *Buffer* parameter.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fileDesc

The file descriptor identifying the file for which exact status information is requested.

Buffer A pointer to the **stat64** structure in which the information is returned. The **stat64** structure is described in the **sys/stat.h** file.

Exit status

If the **gpfs_fstat()** subroutine is successful, it returns a value of 0.

If the **gpfs_fstat()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EBADF

The file descriptor is not valid.

EINVAL

The file descriptor does not refer to a GPFS file or a regular file.

ENOSYS

The **gpfs_fstat()** subroutine is not supported under the current file system format.

| **ESTALE**
| The cached file system information was not valid.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_get_fsnam_from_fssnaphandle() Subroutine

Obtains the file system's name from a GPFS file system snapshot handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_fsnam_from_fssnaphandle
    (gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The **gpfs_get_fsnam_from_fssnaphandle()** subroutine returns a pointer to the name of file system that is uniquely identified by the file system snapshot handle.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

Exit status

If the **gpfs_get_fsnam_from_fssnaphandle()** subroutine is successful, it returns a pointer to the name of the file system identified by the file system snapshot handle.

If the **gpfs_get_fsnam_from_fssnaphandle()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_get_fsnam_from_fssnaphandle()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

GPFS_E_INVALID_FSSNAPHANDLE

The file system snapshot handle is not valid.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_get_fssnaphandle_by_fssnapid() Subroutine

Obtains a GPFS file system snapshot handle given its permanent, unique snapshot ID.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_fssnapid
(const gpfs_fssnap_id_t *fssnapId);
```

Description

The **gpfs_get_fssnaphandle_by_fssnapid()** subroutine creates a handle for the file system or snapshot that is uniquely identified by the permanent, unique snapshot ID.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fssnapId

File system snapshot ID

Exit status

If the **gpfs_get_fssnaphandle_by_fssnapid()** subroutine is successful, it returns a pointer to the file system snapshot handle.

If the **gpfs_get_fssnaphandle_by_fssnapid()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOMEM

Space could not be allocated for the file system snapshot handle.

ENOSYS

The **gpfs_get_fssnaphandle_by_fssnapid()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

GPFS_E_INVALID_FSSNAPID

The file system snapshot ID is not valid.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_get_fssnaphandle_by_name() Subroutine

Obtains a GPFS file system snapshot handle given the file system and snapshot names.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_name
    (const char *fsName, const char *snapName);
```

Description

The **gpfs_get_fssnaphandle_by_name()** subroutine creates a handle for the file system or snapshot that is uniquely identified by the file system's name and the name of the snapshot.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

fsName

A pointer to the name of the file system whose snapshot handle is desired.

snapName

A pointer to the name of the snapshot whose snapshot handle is desired, or NULL to access the active file system rather than a snapshot within the file system.

Exit status

If the **gpfs_get_fssnaphandle_by_name()** subroutine is successful, it returns a pointer to the file system snapshot handle.

If the **gpfs_get_fssnaphandle_by_name()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOENT

The file system name is not valid.

ENOMEM

Space could not be allocated for the file system snapshot handle.

ENOSYS

The **gpfs_get_fssnaphandle_by_name()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

GPFS_E_INVALID_SNAPNAME

The snapshot name is not valid.

Examples

For an example using `gpfs_get_fssnaphandle_by_name()`, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnaphandle_by_path() Subroutine

Obtains a GPFS file system snapshot handle given a path to the file system or snapshot.

Name

gpfs_get_fssnaphandle_by_path() –

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_fssnap_handle_t *gpfs_get_fssnaphandle_by_path
    (const char *pathName);
```

Description

The **gpfs_get_fssnaphandle_by_path()** subroutine creates a handle for the file system or snapshot that is uniquely identified by a path through the file system's mount point to a file or directory within the file system or snapshot.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

pathName

A pointer to the path name to a file or directory within the desired file system or snapshot.

Exit status

If the **gpfs_get_fssnaphandle_by_path()** subroutine is successful, it returns a pointer to the file system snapshot handle.

If the **gpfs_get_fssnaphandle_by_path()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOENT

The path name is not valid.

ENOMEM

Space could not be allocated for the file system snapshot handle.

ENOSYS

The **gpfs_get_fssnaphandle_by_path()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

Examples

For an example using `gpfs_get_fssnaphandle_by_path()`, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_fssnapid_from_fssnaphandle() Subroutine

Obtains the permanent, unique GPFS file system snapshot ID given its handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_get_fssnapid_from_fssnaphandle
    (gpfs_fssnap_handle_t *fssnapHandle,
     gpfs_fssnap_id_t *fssnapId);
```

Description

The **gpfs_get_fssnapid_from_fssnaphandle()** subroutine obtains the permanent, globally unique file system snapshot ID of the file system or snapshot identified by the open file system snapshot handle.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

fssnapId

File system snapshot ID.

Exit status

If the **gpfs_get_fssnapid_from_fssnaphandle()** subroutine is successful, it returns a pointer to the file system snapshot ID.

If the **gpfs_get_fssnapid_from_fssnaphandle()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EFAULT

Size mismatch for **fssnapId**.

EINVAL

NULL pointer given for returned **fssnapId**.

ENOSYS

The **gpfs_get_fssnapid_from_fssnaphandle()** subroutine is not available.

GPFS_E_INVAL_FSSNAPHANDLE

The file system snapshot handle is not valid.

Examples

For an example using `gpfs_get_fssnapid_from_fssnaphandle()`, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_pathname_from_fssnaphandle() Subroutine

Obtains the path name of a GPFS file system snapshot given its handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_pathname_from_fssnaphandle
    (gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The **gpfs_get_pathname_from_fssnaphandle()** subroutine obtains the path name of the file system or snapshot identified by the open file system snapshot handle.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

Exit status

If the **gpfs_get_pathname_from_fssnaphandle()** subroutine is successful, it returns a pointer to the path name of the file system or snapshot.

If the **gpfs_get_pathname_from_fssnaphandle()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_get_pathname_from_fssnaphandle()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

GPFS_E_INVALID_FSSNAPHANDLE

The file system snapshot handle is not valid.

Examples

For an example using **gpfs_get_pathname_from_fssnaphandle()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_get_snapdirname() Subroutine

Obtains the name of the directory containing snapshots.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_get_snapdirname(gpfs_fssnap_handle_t *fssnapHandle,
                        char *snapdirName, int bufLen);
```

Description

The **gpfs_get_snapdirname()** subroutine obtains the name of the directory that is used to contain snapshots.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

snapdirName

Buffer into which the name of the snapshot directory will be copied.

bufLen

The size of the provided buffer.

Exit status

If the **gpfs_get_snapdirname()** subroutine is successful, it returns a value of 0 and the **snapdirName** and **bufLen** parameters are set as described above.

If the **gpfs_get_snapdirname()** subroutine is unsuccessful, it returns a value of -1 and the global error variable **errno** is set to indicate the nature of the error.

Exceptions

None.

Error status

ENOMEM

Unable to allocate memory for this request.

ENOSYS

The **gpfs_get_snapdirname()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The buffer is too small to return the snapshot directory name.

ESTALE

The cached file system information was not valid.

GPFS_E_INVAL_FSSNAPHANDLE

The file system snapshot handle is not valid.

Examples

For an example using `gpfs_get_snapdirname()`, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_get_snapname_from_fssnaphandle() Subroutine

Obtains the name of the snapshot identified by the GPFS file system snapshot handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
const char *gpfs_get_snapname_from_fssnaphandle
    (gpfs_fssnap_handle_t *fssnapHandle);
```

Description

The **gpfs_get_snapname_from_fssnaphandle()** subroutine obtains a pointer to the name of a GPFS snapshot given its file system snapshot handle. If the **fssnapHandle** identifies an active file system, as opposed to a snapshot of a file system, **gpfs_get_snapname_from_fssnaphandle()** returns a pointer to a zero-length snapshot name and a successful return code.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

Exit status

If the **gpfs_get_snapname_from_fssnaphandle()** subroutine is successful, it returns a pointer to the name of the snapshot.

If the **gpfs_get_snapname_from_fssnaphandle()** subroutine is unsuccessful, it returns NULL and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_get_snapname_from_fssnaphandle()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

GPFS_E_INVALID_FSSNAPHANDLE

The file system snapshot handle is not valid.

GPFS_E_INVALID_SNAPNAME

The snapshot has been deleted.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_getacl() Subroutine

Retrieves the access control information for a GPFS file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_getacl(char *pathname, int flags, void *aclP);
```

Description

The **gpfs_getacl()** subroutine, together with the **gpfs_putacl()** subroutine, is intended for use by a backup program to save (**gpfs_getacl()**) and restore (**gpfs_putacl()**) the ACL information for the file.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

pathname

The path identifying the file for which the ACLs are being obtained.

flags Consists of one of these values:

- 0** Indicates that the **aclP** parameter is to be mapped with the **gpfs_opaque_acl_t** struct.
The **gpfs_opaque_acl_t** struct should be used by backup and restore programs.

GPFS_GETACL_STRUCT

Indicates that the **aclP** parameter is to be mapped with the **gpfs_acl_t** struct.

The **gpfs_acl_t** struct is provided for applications that need to interpret the ACL.

aclP Pointer to a buffer mapped by the structure **gpfs_opaque_acl_t** or **gpfs_acl_t**, depending on the value of **flags**.

The first four bytes of the buffer must contain its total size.

Exit status

If the **gpfs_getacl()** subroutine is successful, it returns a value of 0.

If the **gpfs_getacl()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EINVAL

The path name does not refer to a GPFS file or a regular file.

| ENOMEM

| Unable to allocate memory for request.

| **ENOTDIR**

| File is not a directory.

| **ENOSPC**

| The buffer is too small to return the entire ACL. The required buffer size is returned in the first four bytes of the buffer pointed to by **aclP**.

ENOSYS

The **gpfs_getacl()** subroutine is not supported under the current file system format.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_iattr_t Structure

Contains attributes of a GPFS inode.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_iattr
{
    int            ia_version;    /* this struct version */
    int            ia_reclen;    /* sizeof this structure */
    int            ia_checksum;   /* validity check on iattr struct */
    gpfs_mode_t    ia_mode;      /* access mode */
    gpfs_uid_t     ia_uid;       /* owner uid */
    gpfs_gid_t     ia_gid;       /* owner gid */
    gpfs_ino_t     ia_inode;     /* file inode number */
    gpfs_gen_t     ia_gen;       /* inode generation number */
    short          ia_nlink;     /* number of links */
    short          ia_flags;     /* flags (defined below) */
    int            ia_blocksize;  /* preferred block size for io */
    gpfs_mask_t    ia_mask;      /* initial attribute mask-not used*/
    gpfs_off64_t   ia_size;      /* file size in bytes */
    gpfs_off64_t   ia_blocks;    /*512 byte blocks of disk held by file*/
    gpfs_timestruc_t ia_atime;    /* time of last access */
    gpfs_timestruc_t ia_mtime;    /* time of last data modification */
    gpfs_timestruc_t ia_ctime;    /* time of last status change */
    gpfs_dev_t     ia_rdev;      /* ID of device */
    int            ia_xperm;     /*non-zero if file has extended acl*/
    int            ia_modsnapid; /*Internal snapshot ID indicating
                                the last time that the file was modified*/
    unsigned int   ia_filesetid; /* fileset ID */
    unsigned int   ia_datapoolid; /* storage pool ID for data */
} gpfs_iattr_t;

/* Define flags for inode attributes */
#define GPFS_IAFLAG_SNAPDIR    0x0001 /* (obsolete) */
#define GPFS_IAFLAG_USRQUOTA   0x0002 /*inode is a user quota file*/
#define GPFS_IAFLAG_GRPQUOTA   0x0004 /*inode is a group quota file*/
#define GPFS_IAFLAG_ERROR      0x0008 /* error reading inode */

/* Define flags for inode replication attributes */
#define GPFS_IAFLAG_REPLMETA    0x0200 /* metadata replication set */
#define GPFS_IAFLAG_REPLDATA    0x0400 /* data replication set */
#define GPFS_IAFLAG_EXPOSED     0x0800 /*may have data on
                                suspended disks*/
#define GPFS_IAFLAG_ILLREPLICATED 0x1000
                                /*maybe not properly replicated*/
#define GPFS_IAFLAG_UNBALANCED  0x2000
                                /*maybe not properly balanced*/
#define GPFS_IAFLAG_DATAUPDATEMISS 0x4000
                                /* has stale data blocks on unavailable disk */
#define GPFS_IAFLAG_METAUPDATEMISS 0x8000
                                /* has stale metadata on unavailable disk */
#define GPFS_IAFLAG_ILLPLACED    0x0100
                                /* may not be properly placed */
#define GPFS_IAFLAG_FILESET_ROOT 0x0010
                                /* root dir of a fileset */
#define GPFS_IAFLAG_NO_SNAP_RESTORE 0x0020
                                /* don't restore from snapshots */
#define GPFS_IAFLAG_FILESETQUOTA 0x0040
                                /* inode is a fileset quota file */

/* Define flags for extended attributes */
```



```

#define GPFS_IAXPERM_ACL      0x0001 /* file has acls */
#define GPFS_IAXPERM_XATTR   0x0002 /* file has extended attributes */
#define GPFS_IAXPERM_DMATTR  0x0004 /* file has dm attributes */
#define GPFS_IAXPERM_DOSATTR 0x0008 /* file has non-default dos attrs */
#define GPFS_IAXPERM_RPATTR  0x0010 /* file has restore policy attrs */

```

Description

The **gpfs_iattr_t** structure contains the various attributes of a GPFS inode.

Members

ia_version

The version number of this structure.

ia_reclen

The size of this structure.

ia_checksum

The checksum for this **gpfs_iattr** structure.

ia_mode

The access mode for this inode.

ia_uid The owner user ID for this inode.

ia_gid

The owner group ID for this inode.

ia_inode

The file inode number.

ia_gen

The inode generation number.

ia_nlink

The number of links for this inode.

ia_flags

The flags (defined above) for this inode.

ia_blocksize

The preferred block size for I/O.

ia_mask

The initial attribute mask (not used).

ia_size

The file size in bytes.

ia_blocks

The number of 512 byte blocks of disk held by the file.

ia_atime

The time of last access.

ia_mtime

The time of last data modification.

ia_ctime

The time of last status change.

ia_rdev

The ID of the device.

ia_xperm

Indicator - nonzero if file has extended ACL.

ia_modsnapid

Internal snapshot ID indicating the last time that the file was modified. Internal snapshot IDs for the current snapshots are displayed by the **mmlssnapshot** command.

ia_filesetid

The fileset ID for the inode.

ia_datapoolid

The storage pool ID for data for the inode.

Examples

For an example using **gpfs_iattr_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_icolse() Subroutine

Closes a file given its inode file handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_icolse(gpfs_ifile_t *ifile);
```

Description

The **gpfs_icolse()** subroutine closes an open file descriptor created by **gpfs_iopen()**.

For an overview of using **gpfs_icolse()** in a backup application, see “Using APIs to develop backup applications” on page 29.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

ifile Pointer to **gpfs_ifile_t** from **gpfs_iopen()**.

Exit status

If the **gpfs_icolse()** subroutine is successful, it returns a value of 0.

If the **gpfs_icolse()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_icolse()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ESTALE

Cached file system information was not valid.

Examples

For an example using **gpfs_icolse()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_ifile_t Structure

Contains a handle for a GPFS inode.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_ifile gpfs_ifile_t;
```

Description

The **gpfs_ifile_t** structure contains a handle for the file of a GPFS inode.

Members

gpfs_ifile

The handle for the file of a GPFS inode.

Examples

For an example using **gpfs_ifile_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_igetattrs() Subroutine

Retrieve all extended file attributes in opaque format.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetattrs(gpfs_ifile_t *ifile,
                  void *buffer,
                  int bufferSize,
                  int *attrSize);
```

Description

The **gpfs_igetattrs()** subroutine retrieves all extended file attributes in opaque format. This subroutine is intended for use by a backup program to save all extended file attributes (ACLs, DMAPI attributes, and so forth) in one invocation. If the file does not have any extended attributes, the subroutine sets **attrSize** to zero.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

ifile Pointer to **gpfs_ifile_t** from **gpfs_iopen()**.

buffer Pointer to buffer for returned attributes.

bufferSize
Size of the buffer.

attrSize
Pointer to returned size of attributes.

Exit status

If the **gpfs_igetattrs()** subroutine is successful, it returns a value of 0.

If the **gpfs_igetattrs()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS
The **gpfs_igetattrs()** subroutine is not available.

EPERM
The caller does not have superuser privileges.

ERANGE
The buffer is too small to return all attributes. Field ***attrSizeP** will be set to the size necessary.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_IFILE

Incorrect **ifile** parameters.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_igetfilesetname() Subroutine

Returns the name of the fileset defined by a fileset ID.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetfilesetname(gpfs_iscan_t *iscan,
                        unsigned int filesetId,
                        void *buffer,
                        int bufferSize);
```

Description

The **gpfs_igetfilesetname()** subroutine is part of the backup by inode interface. The caller provides a pointer to the scan descriptor used to obtain the fileset ID. This library routine will return the name of the fileset defined by the fileset ID. The name is the null-terminated string provided by the administrator when the fileset was defined. The maximum string length is defined by **GPFS_MAXNAMLEN**.

Note:

1. This routine is not thread safe. Only one thread at a time is allowed to invoke this routine for the given scan descriptor.
2. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

iscan Pointer to **gpfs_iscan_t** used to obtain the fileset ID.

filesetId

The fileset ID.

buffer Pointer to buffer for returned attributes.

bufferSize

Size of the buffer.

Exit status

If the **gpfs_igetfilesetname()** subroutine is successful, it returns a value of 0.

If the **gpfs_igetfilesetname()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_igetfilesetname()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The buffer is too small to return all attributes. Field ***attrSizeP** will be set to the size necessary.

Examples

This programming segment gets the fileset name based on the given fileset ID. The returned fileset name is stored in **FileSetNameBuffer**, which has a length of **FileSetNameSize**.

```
gpfs_iscan_t *fsInodeScanP;  
gpfs_igetfilesetname(fsInodeScanP,FileSetId, &FileSetNameBuffer,FileSetNameSize);
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_igetstoragepool() Subroutine

Returns the name of the storage pool for the given storage pool ID.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_igetstoragepool(gpfs_iscan_t *iscan,
                        unsigned int dataPoolId,
                        void *buffer,
                        int bufferSize);
```

Description

The **gpfs_igetstoragepool()** subroutine is part of the backup by inode interface. The caller provides a pointer to the scan descriptor used to obtain the storage pool ID. This routine returns the name of the storage pool for the given storage pool ID. The name is the null-terminated string provided by the administrator when the storage pool was defined. The maximum string length is defined by **GPFS_MAXNAMLEN**.

Note:

1. This routine is not thread safe. Only one thread at a time is allowed to invoke this routine for the given scan descriptor.
2. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

iscan Pointer to **gpfs_iscan_t** used to obtain the storage pool ID.

dataPoolId
The storage pool ID.

buffer Pointer to buffer for returned attributes.

bufferSize
Size of the buffer.

Exit status

If the **gpfs_igetstoragepool()** subroutine is successful, it returns a value of 0.

If the **gpfs_igetstoragepool()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS
The **gpfs_igetstoragepool()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The buffer is too small to return all attributes. Field ***attrSizeP** will be set to the size necessary.

Examples

This programming segment gets the storage pool name based on the given storage pool ID. The returned storage pool name is stored in **StoragePoolNameBuffer** which has the length of **StoragePoolNameSize**.

```
gpfs_iscan_t *fsInodeScanP;  
gpfs_igetstoragepool(fsInodeScanP, StgpoolIdBuffer, &StgpoolNameBuffer, StgpoolNameSize);
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_iopen() Subroutine

Opens a file or directory by inode number.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_ifile_t *gpfs_iopen(gpfs_fssnap_handle_t *fssnapHandle,
                        gpfs_ino_t ino,
                        int open_flags,
                        const gpfs_iattr_t *statxbuf,
                        const char *symlink);
```

Description

The **gpfs_iopen()** subroutine opens a user file or directory for backup. The file is identified by its inode number **ino** within the file system or snapshot identified by the **fssnapHandle**. The **fssnapHandle** parameter must be the same one that was used to create the inode scan that returned the inode number **ino**.

To read the file or directory, the **open_flags** must be set to **GPFS_O_BACKUP**. The **statxbuf** and **symlink** parameters are reserved for future use and must be set to NULL.

For an overview of using **gpfs_iopen()** in a backup application, see “Using APIs to develop backup applications” on page 29.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

ino inode number.

open_flags

GPFS_O_BACKUP

Read files for backup.

O_RDONLY

For **gpfs_iread()**.

statxbuf

This parameter is reserved for future use and should always be set to NULL.

symlink

This parameter is reserved for future use and should always be set to NULL.

Exit status

If the **gpfs_iopen()** subroutine is successful, it returns a pointer to the inode’s file handle.

If the **gpfs_iopen()** subroutine is unsuccessful, it returns NULL and the global error variable **errno** is set to indicate the nature of the error.

Exceptions

None.

Error status

EINVAL

Missing or incorrect parameter.

ENOMEM

Unable to allocate memory for request.

ENOSYS

The **gpfs_iopen()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ESTALE

Cached file system information was not valid.

Examples

For an example using **gpfs_iopen()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_iread() Subroutine

Reads a file opened by **gpfs_iopen()**.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_iread(gpfs_ifile_t *ifile,
               void *buffer,
               int bufferSize,
               gpfs_off64_t *offset);
```

Description

The **gpfs_iread()** subroutine reads data from the file indicated by the **ifile** parameter returned from **gpfs_iopen()**. This subroutine reads data beginning at parameter **offset** and continuing for **bufferSize** bytes into the buffer specified by **buffer**. If successful, the subroutine returns a value that is the length of the data read, and sets parameter **offset** to the offset of the next byte to be read. A return value of 0 indicates end-of-file.

For an overview of using **gpfs_iread()** in a backup application, see “Using APIs to develop backup applications” on page 29.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

ifile Pointer to **gpfs_ifile_t** from **gpfs_iopen()**.

buffer Buffer for the data to be read.

bufferSize

Size of the buffer (that is, the amount of data to be read).

offset Offset of where within the file to read. If **gpfs_iread()** is successful, **offset** is updated to the next byte after the last one that was read.

Exit status

If the **gpfs_iread()** subroutine is successful, it returns the number of bytes read.

If the **gpfs_iread()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EISDIR

The specified file is a directory.

ENOSYS

The **gpfs_iread()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_IFILE

Incorrect **ifile** parameter.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_ireaddir() Subroutine

Reads the next directory entry.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireaddir(gpfs_ifile_t *idir,
                  const gpfs_direntx_t **dirent);
```

Description

The **gpfs_ireaddir()** subroutine returns the next directory entry in a file system. For an overview of using **gpfs_ireaddir()** in a backup application, see “Using APIs to develop backup applications” on page 29.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

idir Pointer to **gpfs_ifile_t** from **gpfs_iopen()**.

dirent Pointer to returned pointer to directory entry.

Exit status

If the **gpfs_ireaddir()** subroutine is successful, it returns a value of 0 and sets the **dirent** parameter to point to the returned directory entry. If there are no more GPFS directory entries, **gpfs_ireaddir()** returns a value of 0 and sets the **dirent** parameter to NULL.

If the **gpfs_ireaddir()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOMEM

Unable to allocate memory for request.

ENOSYS

The **gpfs_ireaddir()** subroutine is not available.

ENOTDIR

File is not a directory.

EPERM

The caller does not have superuser privileges.

ESTALE

The cached file system information was not valid.

GPFS_E_INVAL_IFILE

Incorrect **ifile** parameter.

Examples

For an example using `gpfs_ireaddir()`, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_ireadlink() Subroutine

Reads a symbolic link by inode number.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_ireadlink(gpfs_fssnap_handle_t *fssnapHandle,
                  gpfs_ino_t ino,
                  char *buffer,
                  int bufferSize);
```

Description

The **gpfs_ireadlink()** subroutine reads a symbolic link by inode number. Like **gpfs_iopen()**, use the same **fssnapHandle** parameter that was used by the inode scan.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

ino inode number of the link file to read.

buffer Pointer to buffer for the returned link data.

bufferSize

Size of the buffer.

Exit status

If the **gpfs_ireadlink()** subroutine is successful, it returns the number of bytes read.

If the **gpfs_ireadlink()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_ireadlink()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The buffer is too small to return the symbolic link.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_FSNAPHANDLE

The file system snapshot handle is not valid.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_ireadx() Subroutine

Performs block level incremental read of a file within an incremental inode scan.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_off64_t gpfs_ireadx(gpfs_ifile_t *ifile,
                        gpfs_iscan_t *iscan,
                        void *buffer,
                        int bufferSize,
                        gpfs_off64_t *offset,
                        gpfs_off64_t termOffset,
                        int *hole);
```

Description

The **gpfs_ireadx()** subroutine performs a block level incremental read on a file opened by **gpfs_iopen()** within a given incremental scan opened using **gpfs_open_inodescan()**.

For an overview of using **gpfs_ireadx()** in a backup application, see “Using APIs to develop backup applications” on page 29.

The **gpfs_ireadx()** subroutines returns the data that has changed since the **prev_fssnapId** specified for the inode scan. The file is scanned starting at **offset** and terminating at **termOffset**, looking for changed data. Once changed data is located, the **offset** parameter is set to its location, the new data is returned in the **buffer** provided, and the amount of data returned is the subroutine’s value.

If the change to the data is that it has been deleted (that is, the file has been truncated), no data is returned, but the **hole** parameter is returned with a value of 1, and the size of the **hole** is returned as the subroutine’s value. The returned size of the hole may exceed the **bufferSize** provided. If no changed data was found before reaching the **termOffset** or the end-of-file, then the **gpfs_ireadx()** subroutine return value is 0.

Block level incremental backups are available only if the previous snapshot was not deleted. If it was deleted, **gpfs_ireadx()** may still be used, but it returns all of the file’s data, operating like the standard **gpfs_iread()** subroutine. However, the **gpfs_ireadx()** subroutine will still identify sparse files and explicitly return information on holes in the files, rather than returning the NULL data.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

ifile Pointer to **gpfs_ifile_t** returned from **gpfs_iopen()**.

iscan Pointer to **gpfs_iscan_t** from **gpfs_open_inodescan()**.

buffer Pointer to buffer for returned data, or NULL to query the next increment to be read.

bufferSize

Size of buffer for returned data.

offset On input, the offset to start the scan for changes. On output, the offset of the changed data, if any was detected.

termOffset

Read terminates before reading this offset. The caller may specify **ia_size** from the file's **gpfs_iattr_t** or 0 to scan the entire file.

hole Pointer to a flag returned to indicated a hole in the file. A value of 0 indicates that the **gpfs_ireadx()** subroutine returned data in the **buffer**. A value of 1 indicates that **gpfs_ireadx()** encountered a hole at the returned **offset**.

Exit status

If the **gpfs_ireadx()** subroutine is successful, it returns the number of bytes read and returned in **bufP**, or the size of the hole encountered in the file.

If the **gpfs_ireadx()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EDOM The file system snapshot ID from the **iscanId** does not match the **ifile**'s.

EINVAL

Missing or incorrect parameter.

EISDIR

The specified file is a directory.

ENOMEM

Unable to allocate memory for request.

ENOSYS

The **gpfs_ireadx()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The file system snapshot ID from the **iscanId** is more recent than the **ifile**'s.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_IFILE

Incorrect **ifile** parameter.

GPFS_E_INVALID_ISCAN

Incorrect **iscan** parameter.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_iscan_t Structure

Contains a handle for an inode scan of a GPFS file system or snapshot.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_iscan gpfs_iscan_t;
```

Description

The **gpfs_iscan_t** structure contains a handle for an inode scan of a GPFS file system or snapshot.

Members

gpfs_iscan

The handle for an inode scan for a GPFS file system or snapshot.

Examples

For an example using **gpfs_iscan_t**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_next_inode() Subroutine

Retrieves the next inode from the inode scan.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_next_inode(gpfs_iscan_t *iscan,
                   gpfs_ino_t termIno,
                   const gpfs_iattr_t **iattr);
```

Description

The **gpfs_next_inode()** subroutine obtains the next inode from the specified inode scan and sets the **iattr** pointer to the inode's attributes. The **termIno** parameter can be used to terminate the inode scan before the last inode in the file system or snapshot being scanned. A value of 0 may be provided to indicate the last inode in the file system or snapshot. If there are no more inodes to be returned before the termination inode, the **gpfs_next_inode()** subroutine returns a value of 0 and the inode's attribute pointer is set to NULL.

For an overview of using **gpfs_next_inode()** in a backup application, see "Using APIs to develop backup applications" on page 29.

To generate a full backup, invoke **gpfs_open_inodescan()** with NULL for the **prev_fssnapId** parameter. Repeated invocations of **gpfs_next_inode()** then return inode information about all existing user files, directories and links, in inode number order.

To generate an incremental backup, invoke **gpfs_next_inode()** with the **fssnapId** that was obtained from a **fssnapHandle** at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the **fssnapId** of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For an incremental backup, only inodes of files that have changed since the specified previous snapshot will be returned. Any operation that changes the file's **mtime** or **ctime** is considered a change and will cause the file to be included. Files with no changes to the file's data or file attributes, other than a change to **atime**, are omitted from the scan.

Incremental backups return deleted files, but full backups do not. A deleted file is indicated by the field **ia_nlinks** having a value of 0.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

iscan Pointer to the inode scan handle.

termIno

The inode scan terminates before this inode number. The caller may specify **maxIno** from **gpfs_open_inodescan()** or zero to scan the entire inode file.

iattr Pointer to the returned pointer to the inode's **iattr**.

Exit status

If the **gpfs_next_inode()** subroutine is successful, it returns a value of 0 and a pointer. The pointer points to NULL if there are no more inodes. Otherwise, the pointer points to the returned inode's attributes.

If the **gpfs_next_inode()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOMEM

Unable to allocate memory for request.

ENOSYS

The **gpfs_next_inode()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_ISCAN

Incorrect parameters.

Examples

For an example using **gpfs_next_inode()**, see `/usr/lpp/mmfs/samples/util/tsbackup.C`.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_opaque_acl_t Structure

Contains buffer mapping for the **gpfs_getacl()** and **gpfs_putacl()** subroutines.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int            acl_buffer_len;
    unsigned short acl_version;
    unsigned char  acl_type;
    char           acl_var_data[1];
} gpfs_opaque_acl_t;
```

Description

The **gpfs_opaque_acl_t** structure contains size, version, and ACL type information for the **gpfs_getacl()** and **gpfs_putacl()** subroutines.

Members

acl_buffer_len

On input, this field must be set to the total length, in bytes, of the data structure being passed to GPFS. On output, this field contains the actual size of the requested information. If the initial size of the buffer is not large enough to contain all of the information, the **gpfs_getacl()** invocation must be repeated with a larger buffer.

acl_version

This field contains the current version of the GPFS internal representation of the ACL. On input to the **gpfs_getacl()** subroutine, set this field to zero.

acl_type

On input to the **gpfs_getacl()** subroutine, set this field to either **GPFS_ACL_TYPE_ACCESS** or **GPFS_ACL_TYPE_DEFAULT**, depending on which ACL is requested. These constants are defined in the **gpfs.h** header file.

acl_var_data

This field signifies the beginning of the remainder of the ACL information.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_open_inodescan() Subroutine

Opens an inode scan of a file system or snapshot.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
gpfs_iscan_t *gpfs_open_inodescan
    (gpfs_fssnap_handle_t *fssnapHandle,
     const gpfs_fssnap_id_t *prev_fssnapId,
     gpfs_ino_t *maxIno);
```

Description

The **gpfs_open_inodescan()** subroutine opens a scan of the inodes in the file system or snapshot identified by the **fssnapHandle** parameter. The scan traverses all user files, directories and links in the file system or snapshot. The scan begins with the user file with the lowest inode number and returns the files in increasing order. The **gpfs_seek_inode()** subroutine may be used to set the scan position to an arbitrary inode. System files, such as the block allocation maps, are omitted from the scan. The file system must be mounted to open an inode scan.

For an overview of using **gpfs_open_inodescan()** in a backup application, see “Using APIs to develop backup applications” on page 29.

To generate a full backup, invoke **gpfs_open_inodescan()** with NULL for the **prev_fssnapId** parameter. Repeated invocations of **gpfs_next_inode()** then return inode information about all existing user files, directories and links, in inode number order.

To generate an incremental backup, invoke **gpfs_open_inodescan()** with the **fssnapId** that was obtained from a **fssnapHandle** at the time the previous backup was created. The snapshot that was used for the previous backup does not need to exist at the time the incremental backup is generated. That is, the backup application needs to remember only the **fssnapId** of the previous backup; the snapshot itself can be deleted as soon as the backup is completed.

For the incremental backup, any operation that changes the file’s **mtime** or **ctime** causes the file to be included. Files with no changes to the file’s data or file attributes, other than a change to **atime**, are omitted from the scan.

A full inode scan (**prev_fssnapId** set to NULL) does not return any inodes of nonexistent or deleted files, but an incremental inode scan (**prev_fssnapId** not NULL) does return inodes for files that have been deleted since the previous snapshot. The inodes of deleted files have a link count of zero.

If the snapshot indicated by **prev_fssnapId** is available, the caller may benefit from the extended read subroutine, **gpfs_ireadx()**, which returns only the changed blocks within the files. Without the previous snapshot, all blocks within the changed files are returned.

Once a full or incremental backup completes, the **new_fssnapId** must be saved in order to reuse it on a subsequent incremental backup. This **fssnapId** must be provided to the **gpfs_open_inodescan()** subroutine, as the **prev_fssnapId** input parameter.

Note:

1. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX

- **libgpfs.so** for Linux

Parameters

fssnapHandle

File system snapshot handle.

prev_fssnapId

Pointer to file system snapshot ID or NULL. If **prev_fssnapId** is provided, the inode scan returns only the files that have changed since the previous backup. If the pointer is NULL, the inode scan returns all user files.

maxIno

Pointer to inode number or NULL. If provided, **gpfs_open_inodescan()** returns the maximum inode number in the file system or snapshot being scanned.

Exit status

If the **gpfs_open_inodescan()** subroutine is successful, it returns a pointer to an inode scan handle.

If the **gpfs_open_inodescan()** subroutine is unsuccessful, it returns a NULL pointer and the global error variable **errno** is set to indicate the nature of the error.

Exceptions

None.

Error status

EDOM The file system snapshot ID passed for **prev_fssnapId** is from a different file system.

EINVAL

Incorrect parameters.

ENOMEM

Unable to allocate memory for request.

ENOSYS

The **gpfs_open_inodescan()** subroutine is not available.

EPERM

The caller does not have superuser privileges.

ERANGE

The **prev_fssnapId** parameter is the same as or more recent than **snapId** being scanned.

ESTALE

Cached file system information was not valid.

GPFS_E_INVALID_FSSNAPHANDLE

The file system snapshot handle is not valid.

GPFS_E_INVALID_FSSNAPID

The file system snapshot ID passed for **prev_fssnapId** is not valid.

Examples

For an example using **gpfs_open_inodescan()**, see **/usr/lpp/mmfs/samples/util/tsbackup.C**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_prealloc() Subroutine

Pre-allocates disk storage for a GPFS file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_prealloc(int fileDesc, offset_t StartOffset,
                  offset_t BytesToPrealloc)
```

Description

The **gpfs_prealloc()** subroutine is used to preallocate disk storage for a file that has already been opened, prior to writing data to the file. The pre-allocated disk storage is started at the requested offset, **StartOffset**, and covers at least the number of bytes requested, **BytesToPrealloc**. Allocations are rounded to GPFS sub-block boundaries.

Pre-allocating disk space for a file provides an efficient method for allocating storage without having to write any data. This can result in faster I/O compared to a file which gains disk space incrementally as it grows.

Existing data in the file is not modified. Reading any of the pre-allocated blocks returns zeroes.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

fileDesc

An integer specifying the file descriptor returned by **open()**.

The file designated for preallocation must be opened for writing.

StartOffset

The byte offset into the file at which to begin pre-allocation.

BytesToPrealloc

The number of bytes to be pre-allocated.

Exit status

If the **gpfs_prealloc()** subroutine is successful, it returns a value of 0.

If the **gpfs_prealloc()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error. If **errno** is set to one of the following, some storage may have been pre-allocated:

- | • **EDQUOT**
- | • **ENOSPC**

The only way to tell how much space was actually pre-allocated is to invoke the **stat()** subroutine and compare the reported file size and number of blocks used with their values prior to preallocation.

Exceptions

None.

Error status

EACCES

The file is not opened for writing.

EBADF

The file descriptor is not valid.

EDQUOT

A disk quota has been exceeded

EINVAL

The file descriptor does not refer to a GPFS file or a regular file; a negative value was specified for **StartOffset** or **BytesToPrealloc**.

ENOSPC

The file system has run out of disk space.

ENOSYS

The **gpfs_prealloc()** subroutine is not supported under the current file system format.

Examples

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <gpfs.h>

int rc;
int fileHandle = -1;
char* fileNameP = "datafile";
offset_t startOffset = 0;
offset_t bytesToAllocate = 20*1024*1024; /* 20 MB */

fileHandle = open(fileNameP, O_RDWR|O_CREAT, 0644);
if (fileHandle < 0)
{
    perror(fileNameP);
    exit(1);
}

rc = gpfs_prealloc(fileHandle, startOffset, bytesToAllocate);
if (rc < 0)
{
    fprintf(stderr, "Error %d preallocation at %lld for %lld in %s\n",
        errno, startOffset, bytesToAllocate, fileNameP);
    exit(1);
}
```

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_putacl() Subroutine

Restores the access control information for a GPFS file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_putacl(char *pathname, int flags, void *aclP)
```

Description

The **gpfs_putacl()** subroutine together with the **gpfs_getacl()** subroutine is intended for use by a backup program to save (**gpfs_getacl()**) and restore (**gpfs_putacl()**) the ACL information for the file.

Note:

1. The use of **gpfs_fgetattrs()** and **gpfs_fputattrs()** is preferred.
2. You must have **write** access to the file.
3. Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:
 - **libgpfs.a** for AIX
 - **libgpfs.so** for Linux

Parameters

pathname

Path name of the file for which the ACLs is to be set.

flags Consists of one of these values:

0 Indicates that the **aclP** parameter is to be mapped with the **gpfs_opaque_acl_t** struct.
The **gpfs_opaque_acl_t** struct should be used by backup and restore programs.

GPFS_PUTACL_STRUCT

Indicates that the **aclP** parameter is to be mapped with the **gpfs_acl_t** struct.

The **gpfs_acl_t** struct is provided for applications that need to change the ACL.

aclP Pointer to a buffer mapped by the structure **gpfs_opaque_acl_t** or **gpfs_acl_t**, depending on the value of **flags**.

This is where the ACL data is stored, and should be the result of a previous invocation of **gpfs_getacl()**.

Exit status

If the **gpfs_putacl()** subroutine is successful, it returns a value of 0.

If the **gpfs_putacl()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

| **EINVAL**

| The path name does not refer to a GPFS file or a regular file.

| **ENOMEM**

| Unable to allocate memory for request.

ENOSYS

 The **gpfs_putacl()** subroutine is not supported under the current file system format.

| **ENOTDIR**

| File is not a directory.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_quotactl() Subroutine

Manipulates disk quotas on file systems.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_quotactl(char *pathname, int cmd, int id, void *bufferP);
```

Description

The **gpfs_quotactl()** subroutine manipulates disk quotas. It enables, disables, and manipulates disk quotas for file systems on which quotas have been enabled.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

pathname

Specifies the path name of any file within the mounted file system to which the quota control command is to applied.

cmd Specifies the quota control command to be applied and whether it is applied to a user, group, or fileset quota.

The **cmd** parameter can be constructed using **GPFS_QCMD(qcmd, Type)** contained in **gpfs.h**. The **qcmd** parameter specifies the quota control command. The **Type** parameter specifies one of the following quota types:

- user (**GPFS_USRQUOTA**)
- group (**GPFS_GRPQUOTA**)
- fileset (**GPFS_FILESETQUOTA**)

The valid values for the **qcmd** parameter specified in **gpfs.h** are:

Q_QUOTAON

Enables quotas.

Enables disk quotas for the file system specified by the **pathname** parameter and type specified in **Type**. The **id** and **bufferP** parameters are unused. Root user authority is required to enable quotas.

Q_QUOTAOFF

Disables quotas.

Disables disk quotas for the file system specified by the **pathname** parameter and type specified in **Type**. The **id** and **bufferP** parameters are unused. Root user authority is required to disable quotas.

Q_GETQUOTA

Gets quota limits and usage information.

Retrieves quota limits and current usage for a user, group, or fileset specified by the **id** parameter. The **bufferP** parameter points to a **gpfs_quotaInfo_t** structure to hold the returned information. The **gpfs_quotaInfo_t** structure is defined in **gpfs.h**.

Root authority is required if the **id** value is not the current id (user id for **GPFS_USRQUOTA**, group id for **GPFS_GRPQUOTA**) of the caller.

Q_SETQUOTA

Sets quota limits

Sets disk quota limits for a user, group, or fileset specified by the **id** parameter. The **bufferP** parameter points to a **gpfs_quotaInfo_t** structure containing the new quota limits. The **gpfs_quotaInfo_t** structure is defined in **gpfs.h**. Root user authority is required to set quota limits.

Q_SETUSE

Sets quota usage

Sets disk quota usage for a user, group, or fileset specified by the **id** parameter. The **bufferP** parameter points to a **gpfs_quotaInfo_t** structure containing the new quota usage. The **gpfs_quotaInfo_t** structure is defined in **gpfs.h**. Root user authority is required to set quota usage.

Q_SYNC

Synchronizes the disk copy of a file system quota

Updates the on disk copy of quota usage information for a file system. The **id** and **bufferP** parameters are unused. Root user authority is required to synchronize a file system quota.

id Specifies the user, group, or fileset ID to which the quota control command applies. The **id** parameters is interpreted by the specified quota type.

bufferP

Points to the address of an optional, command-specific data structure that is copied in or out of the system.

Exit status

If the **gpfs_quotactl()** subroutine is successful, it returns a value of 0.

If the **gpfs_quotactl()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EACCES

Search permission is denied for a component of a path prefix.

EFAULT

An invalid **bufferP** parameter is supplied. The associated structure could not be copied in or out of the kernel.

EINVAL

One of the following errors:

- The file system is not mounted.
- Invalid command or quota type.
- Invalid input limits: negative limits or soft limits are greater than hard limits.

ENOENT

No such file or directory.

EPERM

The quota control command is privileged and the caller did not have root user authority.

GPFS_E_NO_QUOTA_INST

The file system does not support quotas. This is the actual **errno** generated by GPFS.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_quotaInfo_t Structure

Contains buffer mapping for the **gpfs_quotactl()** subroutine.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct gpfs_quotaInfo
{
    gpfs_off64_t blockUsage;      /* current block count */
    gpfs_off64_t blockHardLimit; /* absolute limit on disk blks alloc */
    gpfs_off64_t blockSoftLimit; /* preferred limit on disk blks */
    gpfs_off64_t blockInDoubt;   /* distributed shares + "lost" usage for blks */
    int          inodeUsage;      /* current # allocated inodes */
    int          inodeHardLimit;  /* absolute limit on allocated inodes */
    int          inodeSoftLimit;  /* preferred inode limit */
    int          inodeInDoubt;    /* distributed shares + "lost" usage for inodes */
    gpfs_uid_t   quoId;           /* uid, gid or fileset id */
    int          entryType;       /* entry type, not used */
    unsigned int blockGraceTime;  /* time limit for excessive disk use */
    unsigned int inodeGraceTime; /* time limit for excessive inode use */
} gpfs_quotaInfo_t;
```

Description

The **gpfs_quotaInfo_t** structure contains detailed information for the **gpfs_quotactl()** subroutine.

Members

blockUsage

Current block count in 1 KB units.

blockHardLimit

Absolute limit on disk block allocation.

blockSoftLimit

Preferred limit on disk block allocation.

blockInDoubt

Distributed shares and block usage that have not been not accounted for.

inodeUsage

Current number of allocated inodes.

inodeHardLimit

Absolute limit on allocated inodes.

inodeSoftLimit

Preferred inode limit.

inodeInDoubt

Distributed inode share and inode usage that have not been accounted for.

quoId user ID, group ID, or fileset ID.

entryType

Not used

blockGraceTime

Time limit (in seconds since the Epoch) for excessive disk use.

inodeGraceTime

Time limit (in seconds since the Epoch) for excessive inode use.

Epoch is midnight on January 1, 1970 UTC (Coordinated Universal Time).

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfs_seek_inode() Subroutine

Advances an inode scan to the specified inode number.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_seek_inode(gpfs_iscan_t *iscan,
                   gpfs_ino_t ino);
```

Description

The **gpfs_seek_inode()** subroutine advances an inode scan to the specified inode number.

The **gpfs_seek_inode()** subroutine is used to start an inode scan at some place other than the beginning of the inode file. This is useful to restart a partially completed backup or an interrupted dump transfer to a mirror. It could also be used to do an inode scan in parallel from multiple nodes, by partitioning the inode number space into separate ranges for each participating node. The maximum inode number is returned when the scan was opened and each invocation to obtain the next inode specifies a termination inode number to avoid returning the same inode more than once.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

iscan Pointer to the inode scan handle.

ino The next inode number to be scanned.

Exit status

If the **gpfs_seek_inode()** subroutine is successful, it returns a value of 0.

If the **gpfs_seek_inode()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

ENOSYS

The **gpfs_seek_inode()** subroutine is not available.

GPFS_E_INVALID_ISCAN

Incorrect parameters.

Examples

For an example using **gpfs_seek_inode()**, see **/usr/lpp/mmfs/samples/util/tsinode.c**.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfs_stat() Subroutine

Returns exact file status for a GPFS file.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Synopsis

```
#include <gpfs.h>
int gpfs_stat(char *pathName, struct stat64 *Buffer)
```

Description

The **gpfs_stat()** subroutine is used to obtain exact information about the file named by the **pathName** parameter. This subroutine is provided as an alternative to the **stat()** subroutine, which may not provide exact **mtime** and **atime** values. See “Exceptions to Open Group technical standards” on page 393.

read, **write**, or **execute** permission for the named file is not required, but all directories listed in the path leading to the file must be searchable. The file information is written to the area specified by the **Buffer** parameter.

Note: Compile any program that uses this subroutine with the **-lgpfs** flag from the following library:

- **libgpfs.a** for AIX
- **libgpfs.so** for Linux

Parameters

pathName

The path identifying the file for which exact status information is requested.

Buffer A pointer to the **stat64** structure in which the information is returned. The **stat64** structure is described in the **sys/stat.h** file.

Exit status

If the **gpfs_stat()** subroutine is successful, it returns a value of 0.

If the **gpfs_stat()** subroutine is unsuccessful, it returns a value of -1 and sets the global error variable **errno** to indicate the nature of the error.

Exceptions

None.

Error status

EINVAL

The path name does not refer to a GPFS file or a regular file.

| **ENOENT**

| The file does not exist.

ENOSYS

The **gpfs_stat()** subroutine is not supported under the current file system format.

| **ESTALE**

| The cached file system information was not valid.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

gpfsAccessRange_t Structure

Declares an access range within a file for an application.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int          structLen;
    int          structType;
    offset_t     start;
    offset_t     length;
    int          isWrite;
    char         padding[4];
} gpfsAccessRange_t;
```

Description

The **gpfsAccessRange_t** structure declares an access range within a file for an application.

The application accesses file offsets within the given range, and does not access offsets outside the range. Violating this hint may produce worse performance than if no hint was specified.

This hint is useful in situations where a file is partitioned coarsely among several nodes. If the ranges do not overlap, each node can specify which range of the file it accesses. This provides a performance improvement in some cases, such as for sequential writing within a range.

Subsequent **GPFS_ACCESS_RANGE** hints replace a hint passed earlier.

Members

structLen

Length of the **gpfsAccessRange_t** structure.

structType

Structure identifier **GPFS_ACCESS_RANGE**.

start The start of the access range offset, in bytes, from beginning of file.

length Length of the access range.

0 indicates to end of file.

isWrite

0 indicates **read** access.

1 indicates **write** access.

padding[4]

Provided to make the length of the **gpfsAccessRange_t** structure a multiple of 8 bytes in length. There is no need to initialize this field.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsCancelHints_t Structure

Indicates to remove any hints against the open file handle.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int structLen;
    int structType;
} gpfsCancelHints_t;
```

Description

The **gpfsCancelHints_t** structure indicates to remove any hints against the open file handle.

GPFS removes any hints that may have been issued against this open file handle:

- The hint status of the file is restored to what it would have been immediately after being opened, but does not affect the contents of the GPFS file cache. Cancelling an earlier hint that resulted in data being removed from the GPFS file cache does not bring that data back into the cache. Data reenters the cache only upon access by the application or by user-driven or automatic prefetching.
- Only the **GPFS_MULTIPLE_ACCESS_RANGE** hint has a state that might be removed by the **GPFS_CANCEL_HINTS** directive.

Note: This directive cancels only the effect of other hints, not other directives.

Members

structLen

Length of the **gpfsCancelHints_t** structure.

structType

Structure identifier **GPFS_CANCEL_HINTS**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsClearFileCache_t Structure

Indicates file access in the near future is not expected.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int    structLen;
    int    structType;
} gpfsClearFileCache_t;
```

Description

The **gpfsClearFileCache_t** structure indicates file access in the near future is not expected.

The application does not expect to make any further accesses to the file in the near future, so GPFS removes any data or metadata pertaining to the file from its cache.

Multiple node applications that have finished one phase of their computation may want to use this hint before the file is accessed in a conflicting mode from another node in a later phase. The potential performance benefit is that GPFS can avoid later synchronous cache consistency operations.

Members

structLen

Length of the **gpfsClearFileCache_t** structure.

structType

Structure identifier **GPFS_CLEAR_FILE_CACHE**.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsDataShipMap_t Structure

Indicates which agent nodes are to be used for data shipping.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
#define GPFS_MAX_DS_AGENT_NODES 2048
typedef struct
{
    int    structLen;
    int    structType;
    int    partitionSize;
    int    agentCount;
    int    agentNodeNumber[GPFS_MAX_DS_AGENT_NODES]
} gpfsDataShipMap_t;
```

Description

The **gpfsDataShipMap_t** structure indicates which agent nodes are to be used for data shipping.

GPFS recognizes which agent nodes to use for data shipping:

- This directive can only appear in a **gpfs_fcntl()** subroutine that also gives the **GPFS_DATA_SHIP_START** directive.
- If any of the participating threads include an explicit agent mapping with this directive, all threads must provide the same agent mapping, or else GPFS returns **EINVAL** in **errno**. If this directive is not used, the agents are exactly the nodes on which the **GPFS_DATA_SHIP_START** directive was given. The order of these nodes in the mapping is random. Once the order is set, when all instances have issued the **GPFS_DATA_SHIP_START** directive, the partitioning of the blocks is round robin among the agent nodes.
- All of the nodes named in the data shipping mapping must also be data shipping clients that have issued the **GPFS_DATA_SHIP_START** directive. The reason for this is that GPFS, like most file systems, does not guarantee that data is written through to disk immediately after a **write** call from an application, or even after a **close** returns. Thus, cached data can be lost if a node crashes. Data loss can only occur, however, if the node that crashes is the node that wrote the data.

With data shipping, this property is no longer true. Any node crash in the collective of nodes can cause loss of data. An application running with a file in data shipping mode writes data by shipping it to the GPFS cache on an agent node. That agent node may later crash before writing the data to disk. The originating node may not receive, pay attention to, or realize the severity of an error message. Presumably, a distributed application would notice a crash of one of the nodes on which it was running and would take corrective action, such as rolling back to an earlier stable checkpoint or deleting a corrupt output file. By requiring that all agent nodes also have at least one data shipping client, GPFS makes it such that at least one of the nodes of a distributed application will crash if there is the potential for data loss because of an agent node crash. If any of the data shipping client nodes suffers a node or GPFS crash, the file will be taken out of data shipping mode.

The value for **partitionSize** must be a multiple of the number of bytes in a single file system block.

Members

structLen

Length of the **gpfsDataShipMap_t** structure.

structType

Structure identifier **GPFS_DATA_SHIP_MAP**.

partitionSize

The number of contiguous bytes per server.

This value must be a multiple of the number of bytes in a single file system block.

agentCount

The number of entries in the **agentNodeNumber** array.

agentNodeNumber array

The data ship agent node numbers assigned by GPFS and displayed with the **mmlscluster** command.

Error status**EINVAL**

Not all participating threads have provided the same agent mapping.

ENOMEM

The available data space in memory is not large enough to allocate the data structures necessary to run in data shipping mode.

EPERM

An attempt to **open** a file in data shipping mode that is already open in **write** mode by some thread that did not issue the **GPFS_DATA_SHIP_START** directive.

ESTALE

A node in the data shipping collective has gone down.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsDataShipStart_t Structure

Initiates data shipping mode.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int    structLen;
    int    structType;
    int    numInstances;
    int    reserved;
} gpfsDataShipStart_t;
```

Description

The **gpfsDataShipStart_t** structure initiates data shipping mode.

Once all participating threads have issued this directive for a file, GPFS enters a mode where it logically partitions the blocks of the file among a group of agent nodes. The agents are those nodes on which one or more threads have issued the **GPFS_DATA_SHIP_START** directive. Each thread that has issued a **GPFS_DATA_SHIP_START** directive and the associated agent nodes are referred to as the data shipping collective.

In data shipping mode:

- All file accesses result in GPFS messages to the appropriate agents to **read** or **write** the requested data.
- The **GPFS_DATA_SHIP_START** directive is a blocking collective operation. That is, every thread that intends to access the file through data shipping must issue the **GPFS_DATA_SHIP_START** directive with the same value of **numInstances**. These threads all block within their **gpfs_fcntl()** subroutines until all **numInstances** threads have issued the **GPFS_DATA_SHIP_START** directive.
- The number of threads that issue the **GPFS_DATA_SHIP_START** directive does not have to be the same on all nodes. However, each thread must use a different file handle. The default agent mapping can be overridden using the **GPFS_DATA_SHIP_MAP** directive.
- Applications that perform a fine-grained write, sharing across several nodes, should benefit most from data shipping. The reason for this is that the granularity of GPFS cache consistency is an entire file block, which rarely matches the record size of applications. Without using data shipping, when several nodes simultaneously write into the same block of a file, even non-overlapping parts of the block, GPFS serially grants, and then releases, permission to write into the block to each node in turn. Each permission change requires dirty cached data on the relinquishing node to be flushed to disk, yielding poor performance. Data shipping avoids this overhead by moving data to the node that already has permission to write into the block rather than migrating access permission to the node trying to write the data.

However, since most data accesses are remote in data shipping mode, clients do not benefit from caching as much in data shipping mode as they would if data shipping mode were not in effect. The cost to send a message to another instance of GPFS to fetch or write data is much higher than the cost of accessing that data through the local GPFS buffer cache. Thus, whether or not a particular application benefits from data shipping is highly dependent on its access pattern and its degree of block sharing.

- Another case where data shipping can help performance is when multiple nodes must all append data to the current end of the file. If all of the participating threads open their instances with the **O_APPEND** flag before initiating data shipping, one of the participating nodes is chosen as the agent to which all

appends are shipped. The aggregate performance of all the appending nodes is limited to the throughput of a single node in this case, but should still exceed what the performance would have been for appending small records without using data shipping.

Data shipping mode imposes several restrictions on file usage:

- Because an application level **read** or **write** may be split across several agents, POSIX **read** and **write** file atomicity is not enforced while in data shipping mode.
- A file in data shipping mode cannot be written through any file handle that was not associated with the data shipping collective through a **GPFS_DATA_SHIP_START** directive.
- Calls that are not allowed on a file that has data shipping enabled:
 - **chmod**
 - **fchmod**
 - **chown**
 - **fchown**
 - **link**

The **GPFS_DATA_SHIP_START** directive exits cleanly only when cancelled by a **GPFS_DATA_SHIP_STOP** directive. If all threads issue a **close** for the file, it is taken out of data shipping mode but errors are also returned.

Members

structLen

Length of the **gpfsDataShipStart_t** structure.

structType

Structure identifier **GPFS_DATA_SHIP_START**

numInstances

The number of open file instances, on all nodes, collaborating to operate on the file.

reserved

This field is currently not used.

For compatibility with future versions of GPFS, set this field to zero.

Recovery

Since **GPFS_DATA_SHIP_START** directives block their invoking threads until all participants respond accordingly, there needs to be a way to recover if the application program uses the wrong value for **numInstances** or one of the participating nodes crashes before issuing its **GPFS_DATA_SHIP_START** directive. While a **gpfs_fcntl()** subroutine is blocked waiting for other threads, the subroutine can be interrupted by any signal. If a signal is delivered to any of the waiting subroutines, all waiting subroutine on every node are interrupted and return **EINTR**. GPFS does not establish data shipping if such a signal occurs.

It is the responsibility of the application to mask off any signals that might normally occur while waiting for another node in the data shipping collective. Several libraries use **SIGALRM**; the thread that makes the **gpfs_fcntl()** invocation should use **sigthreadmask** to mask off delivery of this signal while inside the subroutine.

Error status

EINTR A signal was delivered to a blocked **gpfs_fcntl()** subroutine. All waiting subroutines, on every node, are interrupted.

EINVAL

The file mode has been changed since the file was opened to include or exclude **O_APPEND**.

The value of **numInstances** is inconsistent with the value issued by other threads intending to access the file.

An attempt has been made to issue a **GPFS_DATA_SHIP_START** directive on a file that is already in use in data shipping mode by other clients.

ENOMEM

The available data space in memory is not large enough to allocate the data structures necessary to establish and/or run in data shipping mode.

EPERM

An attempt has been made to open a file in data shipping mode that is already open in **write** mode by some thread that did not issue the **GPFS_DATA_SHIP_START** directive. GPFS does not initiate data shipping.

ESTALE

A node in the data shipping collective has gone down.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsDataShipStop_t Structure

Takes a file out of data shipping mode.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int    structLen;
    int    structType;
} gpfsDataShipStop_t;
```

Description

The **gpfsDataShipStop_t** structure takes a file out of data shipping mode.

GPFS takes the file out of data shipping mode:

- GPFS waits for all threads that issued the **GPFS_DATA_SHIP_START** directive to issue this directive, then flushes all dirty file data to disk.
- While a **gpfs_fcntl()** invocation is blocked waiting for other threads, the subroutine can be interrupted by any signal. If a signal is delivered to any of the waiting invocations, all waiting subroutines on every node are interrupted and return **EINTR**. GPFS does not cancel data shipping mode if such a signal occurs. It is the responsibility of the application to mask off any signals that might normally occur while waiting for another node in the data shipping collective. Several libraries use **SIGALRM**; the thread that issues the **gpfs_fcntl()** should use **sigthreadmask** to mask off delivery of this signal while inside the subroutine.

Members

structLen

Length of the **gpfsDataShipStop_t** structure.

structType

Structure identifier **GPFS_DATA_SHIP_STOP**.

Error status

EIO An error occurred while flushing dirty data.

EINTR A signal was delivered to a blocked **gpfs_fcntl()** subroutine. All waiting subroutines, on every node, are interrupted.

EINVAL

An attempt has been made to issue the **GPFS_DATA_SHIP_STOP** directive from a node or thread that is not part of this data shipping collective.

An attempt has been made to issue the **GPFS_DATA_SHIP_STOP** directive on a file that is not in data shipping mode.

ESTALE

A node in the data shipping collective has gone down.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsFcntlHeader_t Structure

Contains declaration information for the **gpfs_fcntl()** subroutine.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int    totalLength;
    int    fcntlVersion;
    int    errorOffset;
    int    fcntlReserved;
} gpfsFcntlHeader_t;
```

Description

The **gpfsFcntlHeader_t** structure contains size, version, and error information for the **gpfs_fcntl()** subroutine.

Members

totalLength

This field must be set to the total length, in bytes, of the data structure being passed in this subroutine. This includes the length of the header and all hints and directives that follow the header.

The total size of the data structure *cannot* exceed the value of **GPFS_MAX_FCNTL_LENGTH**, as defined in the header file **gpfs_fcntl.h**. The current value of **GPFS_MAX_FCNTL_LENGTH** is 64 KB.

fcntlVersion

This field must be set to the current version number of the **gpfs_fcntl()** subroutine, as defined by **GPFS_FCNTL_CURRENT_VERSION** in the header file **gpfs_fcntl.h**. The current version number is one.

errorOffset

If an error occurs processing a system call, GPFS sets this field to the offset within the parameter area where the error was detected.

For example,

1. An incorrect version number in the header, would cause **errorOffset** to be set to zero.
2. An error in the first hint following the header would set **errorOffset** to **sizeof(header)**.

If no errors are found, GPFS does not alter this field.

fcntlReserved

This field is currently unused.

For compatibility with future versions of GPFS, set this field to zero.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsFreeRange_t Structure

Undeclares an access range within a file for an application.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    int          structLen;
    int          structType;
    offset_t     start;
    offset_t     length;
} gpfsFreeRange_t;
```

Description

The **gpfsFreeRange_t** structure undeclares an access range within a file for an application.

The application no longer accesses file offsets within the given range. GPFS flushes the data at the file offsets and removes it from the cache.

Multiple node applications that have finished one phase of their computation may want to use this hint before the file is accessed in a conflicting mode from another node in a later phase. The potential performance benefit is that GPFS can avoid later synchronous cache consistency operations.

Members

structLen

Length of the **gpfsFreeRange_t** structure.

structType

Structure identifier **GPFS_FREE_RANGE**.

start The start of the access range offset, in bytes, from beginning of file.

length Length of the access range.

Zero indicates to end of file.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsGetFilesetName_t Structure

Obtains a file's fileset name.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {  
    int structLen;  
    int structType;  
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];  
} gpfsGetFilesetName_t
```

Description

The **gpfsGetFilesetName_t** structure is used to obtain a file's fileset name.

Members

structLen

Length of the **gpfsGetFilesetName_t** structure.

structType

Structure identifier **GPFS_FCNTL_GET_FILESETNAME**.

buffer The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsGetReplication_t Structure

Obtains a file's replication factors.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int metadataReplicas;
    int maxMetadataReplicas;
    int dataReplicas;
    int maxDataReplicas;
    int status;
    int reserved;
} gpfsGetReplication_t
```

Description

The **gpfsGetReplication_t** structure is used to obtain a file's replication factors.

Members

structLen

Length of the **gpfsGetReplication_t** structure.

structType

Structure identifier **GPFS_FCNTL_GET_REPLICATION**.

metadataReplicas

Returns the current number of copies of indirect blocks for the file.

maxMetadataReplicas

Returns the maximum number of copies of indirect blocks for a file.

dataReplicas

Returns the current number of copies of the data blocks for a file.

maxDataReplicas

Returns the maximum number of copies of data blocks for a file.

status Returns the status of the file. Status values defined below.

reserved

Unused, but should be set to 0.

Error status

These values are returned in the **status** field:

GPFS_FCNTL_STATUS_EXPOSED

This file may have some data where the only replicas are on suspended disks; implies some data may be lost if suspended disks are removed.

GPFS_FCNTL_STATUS_ILLREPLICATE

This file may not be properly replicated; that is, some data may have fewer or more than the desired number of replicas, or some replicas may be on suspended disks.

GPFS_FCNTL_STATUS_UNBALANCED

This file may not be properly balanced.

GPFS_FCNTL_STATUS_DATAUPDATEMISS

This file has stale data blocks on at least one of the disks that are marked as unavailable or recovering in the stripe group descriptor.

GPFS_FCNTL_STATUS_METAUPDATEMISS

This file has stale indirect blocks on at least one unavailable or recovering disk.

GPFS_FCNTL_STATUS_ILLPLACED

This file may not be properly placed; that is, some data may be stored in an incorrect storage pool.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsGetSnapshotName_t Structure

Obtains a file's snapshot name.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {  
    int structLen;  
    int structType;  
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];  
} gpfsGetSnapshotName_t
```

Description

The **gpfsGetSnapshotName_t** structure is used to obtain a file's snapshot name. If the file is not part of a snapshot, a zero length snapshot name will be returned.

Members

structLen

Length of the **gpfsGetSnapshotName_t** structure.

structType

Structure identifier **GPFS_FCNTL_GET_SNAPSHOTNAME**.

buffer The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsGetStoragePool_t Structure

Obtains a file's storage pool name.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {  
    int structLen;  
    int structType;  
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];  
} gpfsGetStoragePool_t
```

Description

The **gpfsGetStoragePool_t** structure is used to obtain a file's storage pool name.

Members

structLen

Length of the **gpfsGetStoragePool_t** structure.

structType

Structure identifier **GPFS_FCNTL_GET_STORAGEPOOL**.

buffer The size of the buffer may vary, but must be a multiple of eight. Upon successful completion of the call, the buffer contains a null-terminated character string for the name of the requested object.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsMultipleAccessRange_t Structure

Defines **prefetching** and **write-behind** file access for an application.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct
{
    offset_t    blockNumber; /* data block number to access */
    int         start; /*start of range (from beginning of block)*/
    int         length; /* number of bytes in range */
    int         isWrite; /* 0 - READ access 1 - WRITE access */
    char        padding[4];
} gpfsRangeArray_t;
```

```
typedef struct
{
    int         structLen;
    int         structType;
    int         accRangeCnt;
    int         relRangeCnt;
    gpfsRangeArray_t accRangeArray[GPFS_MAX_RANGE_COUNT];
    gpfsRangeArray_t relRangeArray[GPFS_MAX_RANGE_COUNT];
} gpfsMultipleAccessRange_t;
```

Description

The **gpfsMultipleAccessRange_t** structure defines **prefetching** and **write-behind** access where the application will soon access the portions of the blocks specified in **accRangeArray** and has finished accessing the ranges listed in **relRangeArray**. The size of a block is returned in the **st_blksize** field of the **stat** command, so the offset, **OFF**, of a file is in the block, **OFF/st_blksize**.

- Up to **GPFS_MAX_RANGE_COUNT**, as defined in the header file **gpfs_fcntl.h**, blocks may be given in one multiple access range hint. The current value of **GPFS_MAX_RANGE_COUNT** is eight. Depending on the current load, GPFS may initiate prefetching of some or all of the blocks.
- Each range named in **accRangeArray** that is accepted for prefetching, should eventually be released with an identical entry in **relRangeArray**, or else GPFS will stop prefetching blocks for this file.

Note: Naming a subrange of a block in **relRangeArray** that does not exactly match a past entry in **accRangeArray** has **no** effect, and does not produce an error.

- Applications that make random accesses or regular patterns not recognized by GPFS may benefit from using this hint.

GPFS already recognizes sequential and strided file access patterns. Applications that use such patterns should not need to use this hint, as GPFS automatically recognizes the pattern and performs **prefetching** and **write-behind** accordingly. In fact, using the multiple access range hint in programs having a sequential or strided access pattern may degrade performance due to the extra overhead to process the hints.

Notice that the units of prefetch and release are file blocks, not file offsets. If the application intends to make several accesses to the same block, it will generally get better performance by including the entire range to be accessed in the **GPFS_MULTIPLE_ACCESS_RANGE** hint before actually doing a **read** or **write**. A sample program **gpfsperf**, which demonstrates the use of the **GPFS_MULTIPLE_ACCESS_RANGE** hint, is included in the GPFS product and installed in the **/usr/lpp/mmfs/samples/perf** directory.

Members

structLen

Length of the **gpfsMultipleAccessRange_t** structure.

structType

Structure identifier **GPFS_MULTIPLE_ACCESS_RANGE**.

accRangeCnt

On input, the number of ranges in **accRangeArray**.

On output, the number of processed ranges, the first n , of the given ranges.

relRangeCnt

The number of ranges in **relRangeArray**.

accRangeArray

The ranges of blocks that the application will soon access.

relRangeArray

The ranges of blocks that the application has finished accessing.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsRestripeData_t Structure

Restripes a file's data blocks.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {  
    int structLen;  
    int structType;  
    int options;  
    int reserved;  
} gpfsRestripeData_t
```

Description

The **gpfsRestripeData_t** structure is used to restripe a file's data blocks to updates its replication and migrate its data. The data movement is always done immediately.

Members

structLen

Length of the **gpfsRestripeData_t** structure.

structType

Structure identifier **GPFS_FCNTL_RESTRIPEDATA**.

options

Options for restripe command. See **mmrestripefs** command for complete definitions.

GPFS_FCNTL_RESTRIPEDATA_M

Migrate critical data off of suspended disks.

GPFS_FCNTL_RESTRIPEDATA_R

Replicate data against subsequent failure.

GPFS_FCNTL_RESTRIPEDATA_P

Place file data in assigned storage pool.

GPFS_FCNTL_RESTRIPEDATA_B

Rebalance file data.

reserved

Must be set to 0.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux

gpfsSetReplication_t Structure

Sets a file's replication factors.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int metadataReplicas;
    int maxMetadataReplicas;
    int dataReplicas;
    int maxDataReplicas;
    int errReason;
    int errValue1;
    int errValue2;
    int reserved;
} gpfsSetReplication_t
```

Description

The **gpfsGetReplication_t** structure is used to set a file's replication factors. However, the directive does not cause the file to be restriped immediately. Instead, the caller must append a **gpfsRestripeData_t** directive or invoke an explicit restripe using the **mmrestripefs** or **mmrestripefile** command.

Members

structLen

Length of the **gpfsSetReplication_t** structure.

structType

Structure identifier **GPFS_FCNTL_SET_REPLICATION**.

metadataReplicas

Specifies how many copies of the file system's metadata to create. Enter a value of 1 or 2, but not greater than the value of the **maxMetadataReplicas** attribute of the file. A value of 0 indicates not to change the current value.

maxMetadataReplicas

The maximum number of copies of indirect blocks for a file. Space is reserved in the inode for all possible copies of pointers to indirect blocks. Valid values are 1 and 2, but cannot be less than **DefaultMetadataReplicas**. The default is 1. A value of 0 indicates not to change the current value.

dataReplicas

Specifies how many copies of the file data to create. Enter a value of 1 or 2, but not greater than the value of the **maxDataReplicas** attribute of the file. A value of 0 indicates not to change the current value.

maxDataReplicas

The maximum number of copies of data blocks for a file. Space is reserved in the inode and indirect blocks for all possible copies of pointers to data blocks. Valid values are 1 and 2 but cannot be less than **DefaultDataReplicas**. The default is 1. A value of 0 indicates not to change the current value.

errReason

Returned reason for request failure. Defined below.

errValue1

Returned value depending upon **errReason**.

errValue2

Returned value depending upon **errReason**.

reserved

Unused, but should be set to 0.

Error status

These values are returned in the **errReason** field:

GPFS_FCNTL_ERR_NONE

Command was successful or no reason information was returned.

GPFS_FCNTL_ERR_METADATA_REPLICAS_RANGE

Field **metadataReplicas** is out of range. Fields **errValue1** and **errValue2** contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_MAXMETADATA_REPLICAS_RANGE

Field **maxMetadataReplicas** is out of range. Fields **errValue1** and **errValue2** contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_DATA_REPLICAS_RANGE

Field **dataReplicas** is out of range. Fields **errValue1** and **errValue2** contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_MAXDATA_REPLICAS_RANGE

Field **maxDataReplicas** is out of range. Fields **errValue1** and **errValue2** contain the valid lower and upper range boundaries.

GPFS_FCNTL_ERR_FILE_NOT_EMPTY

An attempt to change **maxMetadataReplicas** or **maxDataReplicas** or both was made on a file that is not empty.

GPFS_FCNTL_ERR_REPLICAS_EXCEED_FGMAX

Field **metadataReplicas**, or **dataReplicas**, or both exceed the number of failure groups. Field **errValue1** contains the maximum number of metadata failure groups. Field **errValue2** contains the maximum number of data failure groups.

Location

/usr/lpp/mmfs/lib/libgpfs.a for AIX, **/usr/lpp/mmfs/lib/libgpfs.so** for Linux.

gpfsSetStoragePool_t Structure

Sets a file's assigned storage pool.

Library

GPFS Library (**libgpfs.a** for AIX, **libgpfs.so** for Linux)

Structure

```
typedef struct {
    int structLen;
    int structType;
    int errReason;
    int errValue1;
    int errValue2;
    int reserved;
    char buffer[GPFS_FCNTL_MAX_NAME_BUFFER];
} gpfsSetStoragePool_t
```

Description

The **gpfsSetStoragePool_t** structure is used to set a file's assigned storage pool. However, the directive does not cause the file data to be migrated immediately. Instead, the caller must append a **gpfsRestripeData_t** directive or invoke an explicit restripe with the **mmrestripefs** or **mmrestripefile** command. The caller must have su or root privileges to change a storage pool assignment.

Members

structLen

Length of the **gpfsSetStoragePool_t** structure.

structType

Structure identifier **GPFS_FCNTL_SET_STORAGEPOOL**.

buffer The name of the storage pool for the file's data. Only user files may be reassigned to different storage pool. System files, including all directories, must reside in the system pool and may not be moved. The size of the buffer may vary, but must be a multiple of eight.

errReason

Returned reason for request failure. Defined below.

errValue1

Returned value depending upon **errReason**.

errValue2

Returned value depending upon **errReason**.

reserved

Unused, but should be set to 0.

Error status

These values are returned in the **errReason** field:

GPFS_FCNTL_ERR_NONE

Command was successful or no reason information was returned.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL

Invalid storage pool name was given.

GPFS_FCNTL_ERR_INVALID_STORAGE_POOL_TYPE

Invalid storage pool. File cannot be assigned to given pool.

Location

`/usr/lpp/mmfs/lib/libgpfs.a` for AIX, `/usr/lpp/mmfs/lib/libgpfs.so` for Linux

Chapter 10. GPFS user exits

There are three GPFS user exits: **mmsdrbackup**, **nsddevices**, and **syncfsconfig**.

Table 9 summarizes the GPFS-specific user exits.

Table 9. GPFS user exits

User exit	Purpose
"mmsdrbackup User exit" on page 390	Performs a backup of the GPFS configuration data.
"nsddevices User exit" on page 391	Identifies local physical devices that are used as GPFS NSDs.
"syncfsconfig User exit" on page 392	Keeps file system configuration data in replicated clusters synchronized.

mmsdrbackup User exit

Performs a backup of the GPFS configuration data.

Description

The **/var/mmfs/etc/mmsdrbackup** user exit, when properly installed on the primary GPFS configuration server, will be asynchronously invoked every time there is a change to the GPFS master configuration file. This user exit can be used to create a backup of the GPFS configuration data.

Read the sample file **/usr/lpp/mmfs/samples/mmsdrbackup.sample** for a detailed description on how to code and install this user exit.

Parameters

The generation number of the most recent version of the GPFS configuration data.

Exit status

The **mmsdrbackup** user exit should always returns a value of zero.

Location

/var/mmfs/etc

nsddevices User exit

Identifies local physical devices that are used as GPFS Network Shared Disks (NSDs).

Description

The **/var/mmfs/etc/nsddevices** user exit, when properly installed, is invoked synchronously by the GPFS daemon during its disk discovery processing. The purpose of this procedure is to discover and verify the physical devices on each node that correspond to the disks previously defined to GPFS with the **mmcrnsd** command. The **nsddevices** user exit can be used to either replace or to supplement the disk discovery procedure of the GPFS daemon.

Read the sample file **/usr/lpp/mmfs/samples/nsddevices.sample** for a detailed description on how to code and install this user exit.

Parameters

None.

Exit status

The **nsddevices** user exit should return either zero or one.

When the **nsddevices** user exit returns a value of zero, the GPFS disk discovery procedure is bypassed.

When the **nsddevices** user exit returns a value of one, the GPFS disk discovery procedure is performed and the results are concatenated with the results from the **nsddevices** user exit.

Location

/var/mmfs/etc

syncfsconfig User exit

Keeps file system configuration data in replicated clusters synchronized.

Description

The **/var/mmfs/etc/syncfsconfig** user exit, when properly installed, will be synchronously invoked after each command that may change the configuration of a file system. Examples of such commands are: **mmadddisk**, **mmdeldisk**, **mmchfs**, and so forth. The **syncfsconfig** user exit can be used to keep the file system configuration data in replicated GPFS clusters automatically synchronized.

Read the sample file **/usr/lpp/mmfs/samplelessyncfsconfig.sample** for a detailed description on how to code and install this user exit.

Parameters

None.

Exit status

The **syncfsconfig** user exit should always return a value of zero.

Location

/var/mmfs/etc

Chapter 11. Considerations for GPFS applications

Application design should take into consideration the exceptions to Open Group technical standards with regard to the **stat()** system call, and NFS V4 ACLs. Also, a technique to determine if a file system is controlled by GPFS has been provided.

Be sure to consider the following:

- “Exceptions to Open Group technical standards”
- “Determining if a file system is controlled by GPFS”
- “GPFS exceptions and limitations to NFS V4 ACLs” on page 394

Exceptions to Open Group technical standards

GPFS is designed so that most applications written to The Open Group technical standard for file system calls can access GPFS data with no modification, however, there are some exceptions.

Applications that depend on exact reporting of changes to the following fields returned by the **stat()** call may not work as expected:

1. **exact mtime**
2. **mtime**
3. **ctime**
4. **atime**

Providing exact support for these fields would require significant performance degradation to all applications executing on the system. These fields are guaranteed accurate when the file is closed.

These values will be accurate on a node right after it accesses or modifies a file, but may not be accurate for a short while when a file is accessed or modified on some other node.

If 'exact mtime' is specified for a file system (using the **mmcrfs** or **mmchfs** commands with the **-E yes** flag), the **mtime** and **ctime** values are always correct by the time the **stat()** call gives its answer. If 'exact mtime' is not specified, these values will be accurate after a couple of minutes, to allow the synchronization daemons to propagate the values to all nodes. Regardless of whether 'exact mtime' is specified, the **atime** value will be accurate after a couple of minutes, to allow for all the synchronization daemons to propagate changes.

Alternatively, you may use the GPFS calls, **gpfs_stat()** and **gpfs_fstat()** to return exact **mtime** and **atime** values.

The delayed update of the information returned by the **stat()** call also impacts system commands which display disk usage, such as **du** or **df**. The data reported by such commands may not reflect changes that have occurred since the last sync of the file system. For a parallel file system, a sync does not occur until all nodes have individually synchronized their data. On a system with no activity, the correct values will be displayed after the sync daemon has run on all nodes.

Determining if a file system is controlled by GPFS

A file system is controlled by GPFS if the **f_type** field in the **statfs** structure returned from a **statfs()** or **fstatfs()** call has the value 0x47504653, which is the ASCII characters 'GPFS'.

This constant is in the **gpfs.h** file, with the name **GPFS_SUPER_MAGIC**. If an application includes **gpfs.h**, it can compare **f_type** to **GPFS_SUPER_MAGIC** to determine if the file system is controlled by GPFS.

GPFS exceptions and limitations to NFS V4 ACLs

GPFS has exceptions and limitations to the NFS V4 ACLs that you need to understand.

Those exceptions include:

1. Alarm type ACL entries are not supported.
2. Audit type ACL entries are not supported.
3. Inherit entries (**FileInherit** and **DirInherit**) are always propagated to all child subdirectories. The NFS V4 **ACE4_NO_PROPAGATE_INHERIT_ACE** flag is not supported.
4. Although the NFS V4 ACL specification provides separate controls for **WRITE** and **APPEND**, GPFS will not differentiate between the two. Either both must be specified, or neither can be.
5. Similar to **WRITE** and **APPEND**, NFS V4 allows for separate **ADD_FILE** and **ADD_SUBDIRECTORY** controls. In most cases, GPFS will allow these controls to be specified independently. In the special case where the file system object is a directory and one of its ACL entries specifies both **FileInherit** and **DirInherit** flags, GPFS cannot support setting **ADD_FILE** without **ADD_SUBDIRECTORY** (or the other way around). When this is intended, we suggest creating separate **FileInherit** and **DirInherit** entries.
6. Some types of access for which NFS V4 defines controls do not currently exist in GPFS. For these, ACL entries will be accepted and saved, but since there is no corresponding operation they will have no effect. These include **READ_NAMED**, **WRITE_NAMED**, and **SYNCHRONIZE**.
7. AIX requires that **READ_ACL** and **WRITE_ACL** always be granted to the object owner. Although this contradicts *NFS Version 4 Protocol*, it is viewed that this is an area where users would otherwise erroneously leave an ACL that only privileged users could change. Since ACLs are themselves file attributes, **READ_ATTR** and **WRITE_ATTR** are similarly granted to the owner. Since it would not make sense to then prevent the owner from accessing the ACL from a non-AIX node, GPFS has implemented this exception everywhere.
8. AIX does not support the use of special name values other than **owner@**, **group@**, and **everyone@**. Therefore, these are the only valid special name for use in GPFS NFS V4 ACLs as well.
9. NFS V4 allows ACL entries that grant users (or groups) permission to change the owner or owning group of the file (for example, with the **chown** command). For security reasons, GPFS now restricts this so that non-privileged users may only **chown** such a file to themselves (becoming the owner) or to a group that they are a member of.
10. GPFS does not support NFS V4 exporting GPFS file systems from Linux nodes. NFS V3 is acceptable.
11. Windows does not support NFS V4 ACLs.

For more information about GPFS ACLs and NFS export, see *Managing GPFS access control lists and NFS export* in *General Parallel File System: Administration and Programming Reference*.

Chapter 12. File system format changes between versions of GPFS

Every GPFS file system has a format version number associated with it. This version number corresponds to the on-disk data structures for the file system, and is an indicator of the supported file system functionality.

The file system version number is assigned when the file system is first created, and is updated to the latest supported level after the file system is migrated using the **mmchfs -V** command.

The format version number for a file system can be displayed with the **mmlsfs -V** command. If a file system was created with an older GPFS release, new functionality that requires different on-disk data structures will not be enabled until you run the **mmchfs -V** command.

| **Note:** The **-V** option cannot be used to make file systems created prior to GPFS 3.2.1.5 available to
| Windows nodes. Windows nodes can mount only file systems that are created with GPFS 3.2.1.5 or
| later.

The **mmchfs -V** option requires the specification of one of two values - **full** or **compat**:

- Specifying **mmchfs -V full** enables all of the new functionality that requires different on-disk data structures. After this command, nodes in remote clusters running an older GPFS version will no longer be able to mount the file system.
- Specifying **mmchfs -V compat** enables only features that are compatible with nodes running GPFS 3.1. After this command, nodes in remote clusters running GPFS 3.1 will still be able to mount the file system, but nodes running GPFS versions older than 3.1 will not be able to mount the file system.

The current highest file system format version is 10.00. This is the version that is assigned to file systems created with GPFS 3.2. The same version number will be assigned to older file systems after you run the **mmchfs -V** command.

If your current file system is at format level 9.03 (GPFS 3.1), the set of enabled features depends on the value specified with the **mmchfs -V** option:

- After running **mmchfs -V full**, the file system will be able to support:
 - Fine grain directory locking
 - LIMIT clause on placement policies
- After running **mmchfs -V compat**, the file system will be enabled for:
 - Fine grain directory locking (provided all nodes accessing the directory are at GPFS 3.2.1.3 or later)

If your current file system is at format level 8.00 (GPFS 2.3), after running **mmchfs -V**, the file system will be able to support all of the above, plus:

- Storage pools
- Filesets
- Fileset quotas

If your current file system is at format level 7.00 (GPFS 2.2), after running **mmchfs -V**, the file system will be able to support all of the above, plus:

- NFS V4 Access Control Lists
- New format for the internal allocation summary files

If your current file system is at format level 6.00 (GPFS 2.1), after running **mmchfs -V**, the file system will be able to support all of the above, plus extended access control list entries (**-rwx** access mode bits).

The functionality described above is only a subset of the functional changes introduced with the different GPFS releases. Functional changes that do not require changing the on-disk data structures are not listed here. Such changes are either immediately available when the new level of code is installed, or require running the **mmchconfig release=LATEST** command. For a complete list, see the Summary of changes.

Accessibility features for GPFS

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in GPFS:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

The **IBM Cluster Information Center**, and its related publications, are accessibility-enabled. The accessibility features of the information center are described at <http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.addinfo.doc/access.html>.

Keyboard navigation

This product uses standard Microsoft® Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility:

<http://www.ibm.com/able>

Notices

This information was developed for products and services offered in the U.S.A.

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Corporation
Intellectual Property Law
Mail Station P300

2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment or a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interfaces for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

| IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business
| Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked
| terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these
| symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information
| was published. Such trademarks may also be registered or common law trademarks in other countries. A
| current list of IBM trademarks is available on the Web at "Copyright and trademark information" at
| www.ibm.com/legal/copytrade.shtml

Intel®, Intel Inside® (logos), MMX and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat, the Red Hat "Shadow Man" logo, and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc., in the United States and other countries.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in GPFS documentation. If you do not find the term you are looking for, refer to the index of the appropriate book or view the IBM Glossary of Computing Terms, located on the Internet at: <http://www-306.ibm.com/software/globalization/terminology/index.jsp>.

B

block utilization. The measurement of the percentage of used subblocks per allocated blocks.

C

cluster. A loosely-coupled collection of independent systems (nodes) organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster configuration data. The configuration data that is stored on the cluster configuration servers.

cluster manager. The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager is the node with the lowest node number among the quorum nodes that are operating at a particular time.

control data structures. Data structures needed to manage file data and metadata cached in memory. Control data structures include hash tables and link pointers for finding cached data; lock states and tokens to implement distributed locking; and various flags and sequence numbers to keep track of updates to the cached data.

D

Data Management Application Program Interface (DMAPI). The interface defined by the Open Group's XDSM standard as described in the publication *System Management: Data Storage Management (XDSM) API Common Application Environment (CAE) Specification C429*, The Open Group ISBN 1-85912-190-X.

deadman switch timer. A kernel timer that works on a node that has lost its disk lease and has outstanding I/O requests. This timer ensures that the node cannot complete the outstanding I/O requests (which would risk causing file system corruption), by causing a panic in the kernel.

disk descriptor. A definition of the type of data that the disk contains and the failure group to which this disk belongs. See also *failure group*.

disposition. The session to which a data management event is delivered. An individual disposition is set for each type of event from each file system.

disk leasing. A method for controlling access to storage devices from multiple host systems. Any host that wants to access a storage device configured to use disk leasing registers for a lease; in the event of a perceived failure, a host system can deny access, preventing I/O operations with the storage device until the preempted system has reregistered.

domain. A logical grouping of resources in a network for the purpose of common management and administration.

F

failback. Cluster recovery from failover following repair. See also *failover*.

failover. (1) The process of transferring all control of the ESS to a single cluster in the ESS when the other cluster in the ESS fails. See also *cluster*. (2) The routing of all transactions to a second controller when the first controller fails. See also *cluster*. (3) The assumption of file system duties by another node when a node fails.

failure group. A collection of disks that share common access paths or adapter connection, and could all become unavailable through a single hardware failure.

fileset. A hierarchical grouping of files managed as a unit for balancing workload across a cluster.

file-management policy. A set of rules defined in a policy file that GPFS uses to manage file migration and file deletion. See also *policy*.

file-placement policy. A set of rules defined in a policy file that GPFS uses to manage the initial placement of a newly created file. See also *policy*.

file system descriptor. A data structure containing key information about a file system. This information includes the disks assigned to the file system (*stripe group*), the current state of the file system, and pointers to key files such as quota files and log files.

file system descriptor quorum. The number of disks needed in order to write the file system descriptor correctly.

file system manager. The provider of services for all the nodes using a single file system. A file system manager processes changes to the state or description of the file system, controls the regions of disks that are allocated to each node, and controls token management and quota management.

fragment. The space allocated for an amount of data too small to require a full block. A fragment consists of one or more subblocks.

G

GPFS cluster. A cluster of nodes defined as being available for use by GPFS file systems.

GPFS portability layer. The interface module that each installation must build for its specific hardware platform and Linux distribution.

GPFS recovery log. A file that contains a record of metadata activity, and exists for each node of a cluster. In the event of a node failure, the recovery log for the failed node is replayed, restoring the file system to a consistent state and allowing other nodes to continue working.

I

ill-placed file. A file assigned to one storage pool, but having some or all of its data in a different storage pool.

ill-replicated file. A file with contents that are not correctly replicated according to the desired setting for that file. This situation occurs in the interval between a change in the file's replication settings or suspending one of its disks, and the restripe of the file.

indirect block. A block containing pointers to other blocks.

IBM Virtual Shared Disk. The subsystem that allows application programs running on different nodes to access a logical volume as if it were local to each node. The logical volume is local to only one of the nodes (the server node).

inode. The internal structure that describes the individual files in the file system. There is one inode for each file.

J

journaled file system (JFS). A technology designed for high-throughput server environments, which are important for running intranet and other high-performance e-business file servers.

junction.

A special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

K

kernel. The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

L

logical volume. A collection of physical partitions organized into logical partitions, all contained in a single volume group. Logical volumes are expandable and can span several physical volumes in a volume group.

Logical Volume Manager (LVM). A set of system commands, library routines, and other tools that allow the user to establish and control logical volume (LVOL) storage. The LVM maps data between the logical view of storage space and the physical disk drive module (DDM).

M

metadata. A data structures that contain access information about file data. These include: inodes, indirect blocks, and directories. These data structures are not accessible to user applications.

metanode. The one node per open file that is responsible for maintaining file metadata integrity. In most cases, the node that has had the file open for the longest period of continuous time is the metanode.

mirroring. The process of writing the same data to multiple disks at the same time. The mirroring of data protects it against data loss within the database or within the recovery log.

multi-tailed. A disk connected to multiple nodes.

N

namespace. Space reserved by a file system to contain the names of its objects.

Network File System (NFS). A protocol, developed by Sun Microsystems, Incorporated, that allows any host in a network to gain access to another host or netgroup and their file directories.

Network Shared Disk (NSD). A component for cluster-wide disk naming and access.

NSD volume ID. A unique 16 digit hex number that is used to identify and access all NSDs.

node. An individual operating-system image within a cluster. Depending on the way in which the computer system is partitioned, it may contain one or more nodes.

node descriptor. A definition that indicates how GPFS uses a node. Possible functions include: manager node, client node, quorum node, and nonquorum node

node number. A number that is generated and maintained by GPFS as the cluster is created, and as nodes are added to or deleted from the cluster.

node quorum. The minimum number of nodes that must be running in order for the daemon to start.

node quorum with tiebreaker disks. A form of quorum that allows GPFS to run with as little as one quorum node available, as long as there is access to a majority of the quorum disks.

non-quorum node. A node in a cluster that is not counted for the purposes of quorum determination.

P

policy. A list of file-placement and service-class rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy set is active at one time.

policy rule. A programming statement within a policy that defines a specific action to be preformed.

pool. A group of resources with similar characteristics and attributes.

portability. The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

primary GPFS cluster configuration server. In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data.

private IP address. A IP address used to communicate on a private network.

public IP address. A IP address used to communicate on a public network.

Q

quorum node. A node in the cluster that is counted to determine whether a quorum exists.

quota. The amount of disk space and number of inodes assigned as upper limits for a specified user, group of users, or fileset.

quota management. The allocation of disk blocks to the other nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

Redundant Array of Independent Disks (RAID). A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

recovery. The process of restoring access to file system data when a failure has occurred. Recovery can involve reconstructing data or providing alternative routing through a different server.

replication. The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

rule. A list of conditions and actions that are triggered when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups), the requesting client, and the container name associated with the object.

S

SAN-attached. Disks that are physically attached to all nodes in the cluster using Serial Storage Architecture (SSA) connections or using fibre channel switches

secondary GPFS cluster configuration server. In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data in the event that the primary GPFS cluster configuration server fails or becomes unavailable.

Secure Hash Algorithm digest (SHA digest). A character string used to identify a GPFS security key.

Serial Storage Architecture (SSA). An American National Standards Institute (ANSI) standard, implemented by IBM, for a high-speed serial interface that provides point-to-point connection for peripherals, such as storage arrays.

session failure. The loss of all resources of a data management session due to the failure of the daemon on the session node.

session node. The node on which a data management session was created.

Small Computer System Interface (SCSI). An ANSI-standard electronic interface that allows personal computers to communicate with peripheral hardware, such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

snapshot. A copy of changed data in the active files and directories of a file system with the exception of the inode number, which is changed to allow application programs to distinguish between the snapshot and the active files and directories.

source node. The node on which a data management event is generated.

SSA. See Serial Storage Architecture.

stand-alone client. The node in a one-node cluster.

storage area network (SAN). A dedicated storage network tailored to a specific environment, combining servers, storage products, networking products, software, and services.

storage pool. A grouping of storage space consisting of volumes, logical unit numbers (LUNs), or addresses that share a common set of administrative characteristics.

stripe group. The set of disks comprising the storage assigned to a file system.

striping. A storage process in which information is split into blocks (a fixed amount of data) and the blocks are written to (or read from) a series of disks in parallel.

subblock. The smallest unit of data accessible in an I/O operation, equal to one thirty-second of a data block.

system storage pool. A storage pool containing file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks and extended attributes. The **system storage pool** can also contain user data.

T

token management. A system for controlling file access in which each application performing a read or write operation is granted some form of access to a specific block of file data. Token management provides data consistency and controls conflicts. Token management has two components: the token management server, and the token management function.

token management function. A component of token management that requests tokens from the token management server. The token management function is located on each cluster node.

token management server. A component of token management that controls tokens relating to the operation of the file system. The token management server is located at the file system manager node.

twin-tailed. A disk connected to two nodes.

U

user storage pool. A storage pool containing the blocks of data that make up user files.

V

virtual file system (VFS). A remote file system that has been mounted so that it is accessible to the local user.

virtual shared disk. See *IBM Virtual Shared Disk*.

virtual node (vnode). The structure that contains information about a file system object in a virtual file system (VFS).

Index

Special characters

--aix-trace-buffer-size
 changing 283
--tracedev-buffer-size
 changing 283
.rhosts file 1, 68, 71, 75, 84, 90, 98, 102, 107, 112,
114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166,
168, 171, 173, 175, 177, 185, 189, 194, 199, 203,
205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266,
269, 273, 276, 278, 280, 289

A

access ACL 50
access control information 324
 restoring 355
access control lists
 administering 53
 allow type 53
 applying 56
 changing 52, 56
 creating 246
 DELETE 54
 DELETE_CHILD 54
 deleting 52, 56, 160
 deny type 53
 displaying 51, 56, 196
 editing 179
 exceptions 56
 getting 199
 inheritance 53
 limitations 56
 managing 49
 NFS V4 49, 53
 NFS V4 syntax 53
 setting 50, 51, 55
 special names 53
 traditional 49
 translation 55
accessibility features for the GPFS product 397
ACL information 293, 350
 restoring 355
 retrieving 324
activating quota limit checking 45
adding
 disks 32
adding nodes to a GPFS cluster 6
administering
 GPFS file system 1
administration tasks 1
agent node 369
atime 111, 351, 364, 393
attributes
 useNSDserver 39
authentication method 1
automatic mount
 indicating 132

automatic quota activation 262
automount 17
automounter attribute
 changing 93
availability
 disk 36

B

backing up a file system 28
backup applications
 writing 29
backup client 83
backup control information 83
backup server 83
block level incremental backups 345
block size
 affect on maximum mounted file system size 95,
 132
 choosing 132

C

cache 59
changing
 an administration or daemon interface for a
 node 116
 automounter attribute 93
 cipherList attribute 93
 cluster configuration attributes 92
 cnfsMountdPort attribute 93
 cnfsNFSDprocs attribute 93
 cnfsSharedRoot attribute 93
 cnfsVIP attribute 93
 configuration attributes on the mmchconfig
 command 9
 dataStructureDump attribute 93
 defaultMountDir attribute 93
 designation attribute 93
 disk states 37
 maxblocksize attribute 95
 maxFilesToCache attribute 95
 maxMBps attribute 95
 maxStatCache attribute 95
 nsdServerWaitTimeForMount 95
 nsdServerWaitTimeWindowOnMount 95
 pagepool attribute 96
 prefetchThreads attribute 96
 quotas 43
 replication 23
 sidAutoMapRangeLength attribute 96
 sidAutoMapRangeStart attribute 96
 tiebreakerDisks 97
 tracing attributes 282
 uidDomain attribute 97
 unmountOnDiskFail attribute 97
 usePersistentReserve 97

- changing (*continued*)
 - verbsPorts 97
 - verbsRdma 98
 - worker1Threads attribute 98
- checking
 - file systems 20
 - quotas 43
- chmod 50
- cipherList 79
- cipherList attribute
 - changing 93
- client node
 - refresh NSD server 239
- cluster
 - changing configuration attributes 9, 92
 - changing tracing attributes 282
- cluster configuration attributes
 - changing 9
 - displaying 211
- cluster configuration data 185, 192
- cluster configuration server 124, 170, 192
- clustered NFS subsystem
 - using 60
- cnfsMountdPort attribute
 - changing 93
- cnfsNFSDprocs attribute
 - changing 93
- cnfsSharedRoot attribute
 - changing 93
- cnfsVIP attribute
 - changing 93
- commands 63
 - chmod 50
 - mmaddddisk 32, 66, 392
 - mmaddnode 6, 70
 - mmapplypolicy 73
 - mmauth 78
 - mmbackup 28, 29, 83
 - mmchattr 23, 24, 86
 - mmchcluster 8, 89
 - mmchconfig 9, 12, 58, 59, 92
 - mmchdisk 20, 24, 36, 37, 100
 - mmcheckquota 20, 41, 43, 104
 - mmchfileset 107
 - mmchfs 22, 39, 45, 59, 109
 - mmchmgr 114
 - mmchnode 116
 - mmchnsd 38, 120
 - mmchpolicy 122
 - mmcrcluster 5, 124
 - mmcrfileset 128
 - mmcrfs 17, 45, 53, 59, 130
 - mmcrnsd 31, 32, 33, 66, 137, 271, 391, 392
 - mmcrsnapshot 29, 142
 - mmcrvsd 145
 - mmcvnsd 33
 - mmdefedquota 150
 - mmdefquotaoff 152
 - mmdefquotaon 154
 - mmdefragfs 26, 27, 157
 - mmdelacl 52, 53, 56, 160

- commands (*continued*)
 - mmdeldisk 33, 34, 162, 392
 - mmdelfileset 165
 - mmdelfs 19, 168
 - mmdelnode 7, 170
 - mmdelnsd 172
 - mmdelsnapshot 174
 - mmddf 25, 33, 176
 - mmeditACL 52, 53, 55, 56, 179
 - mmedquota 43, 182
 - mmexportfs 185
 - mmfsck 20, 34, 187
 - mmfsctl 192
 - mmgetacl 51, 55, 56, 196
 - mmgetstate 199
 - mmimportfs 202
 - mmlinkfileset 205
 - mmlsattr 23, 207
 - mmlscluster 5, 209
 - mmlsconfig 211
 - mmlsdisk 20, 36, 213
 - mmlsfileset 216
 - mmlsfs 21, 35, 59, 219
 - mmlsmgr 13, 223
 - mmlsmount 19, 225
 - mmlsnsd 31, 227
 - mmlspolicy 230
 - mmlsquota 44, 232
 - mmlssnapshot 235, 328
 - mmmount 17, 39, 237
 - mmnsddiscover 239
 - mmpmon 241
 - mmputacl 50, 51, 53, 55, 56, 246
 - mmquotaoff 45, 249
 - mmquotaon 45, 251
 - mmremotecluster 253
 - mmremotefs 39, 256
 - mmrepquota 46, 259
 - mmrestorefs 262
 - mmrestripe 35
 - mmrestripefile 265
 - mmrestripefs 24, 33, 268
 - completion time 13
 - mmrpldisk 34, 271
 - mmshutdown 14, 275
 - mmsnapdir 277, 308
 - mmstartup 14, 280
 - mmtracectl 282
 - mmumount 18, 19, 285
 - mmunlinkfileset 288
- rcp 68, 71, 75, 84, 90, 98, 102, 107, 112, 114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166, 168, 171, 173, 175, 177, 185, 189, 194, 199, 203, 205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266, 269, 273, 276, 278, 280, 289
- rsh 68, 71, 75, 84, 90, 98, 102, 107, 112, 114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166, 168, 171, 173, 175, 177, 185, 189, 194, 199, 203, 205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266, 269, 273, 276, 278, 280, 289
- common GPFS command principles 2

- concurrent virtual shared disks
 - creating 145
- configuration *see also* cluster 409
- configuration attributes on the mmchconfig command
 - changing 9
- considerations for GPFS applications 393
- contact node 170
- control file permission 50
- creating
 - file systems 130
 - filesets 128
 - quota reports 46
- ctime 351, 393

D

- data replica 111
- data replication
 - changing 23
- data shipping 369
- data shipping mode 371, 374, 375, 376, 378
- dataStructureDump attribute
 - changing 93
- deactivating quota limit checking 45
- default ACL 50
- default quotas 42
 - activating 154
 - deactivating 152
 - editing 150
- defaultMountDir
 - changing 93
- deleting
 - file systems 19
 - filesets 165
 - nodes from a GPFS cluster 7
- deleting links
 - snapshots 277
- deny-write open lock 109
- description
 - dmapiEventTimeout 94
 - dmapiMountEvent 94
 - dmapiMountTimeout 94
 - dmapiSessionFailureTimeout 94
- designation attribute
 - changing 93
- Direct I/O caching policy
 - using 23
- directives
 - description 61
 - subroutine for passing 298
- directory entry
 - reading 341
- DirlInherit 53
- disabling
 - Persistent Reserve 40
- disaster recovery 192
- disk
 - availability 36
 - status 36
- disk access
 - path discovery 239

- disk descriptor 66, 100, 137
- disk discovery 39
- disk parameter
 - changing 100
- disk state
 - changing 37, 100
 - displaying 36
 - suspended 100
- disk storage
 - pre-allocating 353
- disk usage 100, 131, 271
- disks
 - adding 32, 66, 271
 - configuration 213
 - deleting 33, 162
 - displaying information 31
 - displaying state 213
 - ENOSPC 35
 - failure 24
 - fragmentation 26
 - managing 31
 - maximum number 31
 - reducing fragmentation 157
 - replacing 34, 271
 - strict replication 35
- displaying
 - access control lists 51
 - disk fragmentation 26
 - disk states 36
 - disks 31
 - filesets 216
 - NSD belonging to a GPFS cluster 227
 - quotas 44
- DMAPI 112
- dmapiEventTimeout
 - description 94
- dmapiMountEvent attribute
 - description 94
- dmapiMountTimeout attribute
 - description 94
- dmapiSessionFailureTimeout attribute
 - description 94
- dumps
 - storage of information 93

E

- editing
 - default quotas 150
- enabling
 - Persistent Reserve 40
- establishing quotas 43
- exceptions to Open Group technical standards 393
- execute file permission 49
- exporting a GPFS file system 57
- extended ACLs
 - retrieve 301
 - set 303, 305
- extended file attributes 303, 305
 - retrieve 301
 - set 303, 305

F

- failure group 100, 131, 271
- failureDetectionTime attribute
 - changing 95
- FailureGroup 67
- file
 - access control information 321, 324, 355
 - access range 366
 - ACL information 321, 324, 355
 - block level incremental read 345
 - extended attributes 301, 303, 305
 - reading 339
- file access
 - application driven prefetching and write-behind 382
 - canceling 367, 368
 - clearing 368
 - freeing a given range 376
 - gpfs_fcctl() 61
 - gpfsDataShipStart_t 371
 - gpfsDataShipStop_t 374
- file attribute
 - extended 331
 - querying 207
- file cache 368, 369
- file descriptor
 - closing 329
 - opening 337
- file permissions
 - control 50
 - GPFS extension 49
- file status information 310, 364
- file system descriptor quorum 193
- file system manager
 - changing nodes 13, 114
 - designation 93
 - displaying current 223
 - displaying node currently assigned 13
- file system name 312, 314
- file system snapshot handle 296, 308
- file system space
 - querying 176
- file systems
 - access control lists 49
 - adding disks 66
 - AIX export 58
 - attributes
 - changing 22
 - displaying 21
 - backing up 28, 83
 - block size 130
 - change manager node 114
 - changing attributes 109
 - changing attributes for files 86
 - checking 20, 187, 202
 - control request 192
 - controlled by GPFS 393
 - creating 130
 - creating snapshot 142
 - deleting 19, 168
 - deleting disks 162
 - disk fragmentation 26
- file systems (*continued*)
 - displaying attributes 219
 - displaying format version 219
 - exporting 57, 185
 - exporting using NFS 57
 - file system manager
 - displaying 223
 - format version 109
 - formatting 131
 - fragmentation
 - querying 26
 - GPFS control 393
 - handle 308
 - importing 202
 - inconsistencies 187
 - links to snapshots 277
 - Linux export 57
 - listing mounted 225
 - management 17
 - migrating 109
 - mounted file system sizes 95, 132
 - mounting 237
 - mounting on multiple nodes 17
 - moving to another cluster 109, 185
 - mtime value 109
 - NFS export 59
 - NFS V4 export 59
 - querying space 176
 - quotas 154
 - rebalancing 268
 - reducing fragmentation 27, 157
 - remote 78, 253, 256
 - repairing 20, 187
 - restoring with snapshots 262
 - restripe 67
 - restriping 24, 268
 - space, querying 25
 - unmounting 275, 285
 - unmounting on multiple nodes 19
 - which nodes have mounted 19
- FileInherit 53
- files
 - .rhosts 68, 71, 75, 84, 90, 98, 102, 107, 112, 114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166, 168, 171, 173, 175, 177, 185, 189, 194, 199, 203, 205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266, 269, 273, 276, 278, 280, 289
 - dsm.sys 28
 - orphaned 187
 - rebalancing 265
 - restriping 265
- filesets
 - changing 107
 - creating 128
 - deleting 165
 - displaying 216
 - ID 333
 - linking 205
 - name 333
 - quotas 41
 - unlinking 288

FlashCopy image 192
full backup 83, 348, 351

G

genkey 79
GPFS
 stopping 275
GPFS cache 59
GPFS cluster
 adding nodes 6
 changing the GPFS cluster configuration servers 8
 creating 5, 124
 deleting nodes 7
 displaying configuration information 5
 managing 5
GPFS cluster configuration data 185
GPFS cluster configuration information
 displaying 209
GPFS cluster configuration server 170
 changing 89
 primary 90
 secondary 90
GPFS cluster configuration servers
 changing 8
 choosing 124
 displaying 5
GPFS cluster data 137
GPFS commands 63
GPFS configuration data 390, 392
GPFS daemon
 starting 14, 280
 stopping 14, 275
GPFS daemon status 199
GPFS directory entry 297
GPFS file system
 administering 1
GPFS file system snapshot handle 312, 313, 314,
 316, 318, 320, 323
 free 307
GPFS programming interfaces 291
GPFS subroutines 291
GPFS user exits 389
gpfs_acl_t 293
GPFS_ATTRFLAG_IGNORE_POOL
 gpfs_fgetattrs() 301
 gpfs_fputattrs() 303
 gpfs_fputattrswithpathname() 305
GPFS_ATTRFLAG_NO_PLACEMENT
 gpfs_fgetattrs() 301
 gpfs_fputattrs() 303
 gpfs_fputattrswithpathname() 305
gpfs_close_inodescan() 294
gpfs_cmp_fssnapid() 295
gpfs_direntx_t 297
gpfs_fcntl() 298
gpfs_fgetattrs() 301
 GPFS_ATTRFLAG_IGNORE_POOL 301
 GPFS_ATTRFLAG_NO_PLACEMENT 301
gpfs_fputattrs() 303
 GPFS_ATTRFLAG_IGNORE_POOL 303
gpfs_fputattrs() (continued)
 GPFS_ATTRFLAG_NO_PLACEMENT 303
gpfs_fputattrswithpathname() 305
 GPFS_ATTRFLAG_IGNORE_POOL 305
 GPFS_ATTRFLAG_NO_PLACEMENT 305
gpfs_free_fssnaphandle() 307
gpfs_fssnap_handle_t 308
gpfs_fssnap_id_t 309
gpfs_fstat() 310
gpfs_get_fsname_from_fssnaphandle() 312
gpfs_get_fssnaphandle_by_fssnapid() 313
gpfs_get_fssnaphandle_by_name() 314
gpfs_get_fssnaphandle_by_path() 316
gpfs_get_fssnapid_from_fssnaphandle() 318
gpfs_get_pathname_from_fssnaphandle() 320
gpfs_get_snapdirname() 321
gpfs_get_snapname_from_fssnaphandle() 323
gpfs_getacl() 324
gpfs_iattr_t 326
gpfs_iclose 29
gpfs_iclose() 329
gpfs_ifile_t 330
gpfs_iopen 29
gpfs_iopen() 337
gpfs_iread 29
gpfs_iread() 339
gpfs_ireaddir 29
gpfs_ireaddir() 341
gpfs_ireadlink() 343
gpfs_ireadx() 345
gpfs_iscan_t 347
gpfs_next_inode 29
gpfs_next_inode() 348
gpfs_opaque_acl_t 350
gpfs_open_inodescan 29
gpfs_open_inodescan() 351
gpfs_prealloc() 353
gpfs_putacl() 355
gpfs_quotactl() 42, 357
gpfs_quotaInfo_t 360
gpfs_seek_inode() 362
gpfs_stat() 364
gpfsAccessRange_t 366
gpfsCancelHints_t 367
gpfsClearFileCache_t 368
gpfsDataShipMap_t 369
gpfsDataShipStop_t 374
gpfsFcntlHeader_t 375
gpfsFreeRange_t 376
gpfsGetFilesetName_t 377
gpfsGetReplication_t 378
gpfsGetSnapshotName_t 380
gpfsGetStoragePool_t 381
gpfsMultipleAccessRange_t 382
gpfsRestripeData_t 384
gpfsSetReplication_t 385
gpfsSetStoragePool_t 387
grace period
 changing 182
 setting 182
group quota 150, 152, 154, 183, 232, 251, 259

H

High Performance Switch (HPS) 33
hints
 description 61
 subroutine for passing 298
hole 345

I

I/O caching policy
 changing 86
in-doubt value 104, 232, 259
incremental backup 83, 348, 351
inheritance of ACLs 53
InheritOnly 53
inode
 attributes 326
inode file handle 329, 330
inode number 337, 343, 348, 351, 362
inode scan 345, 348, 362
 closing 294
 opening 351
inode scan handle 294, 347
iscan handle 294

K

kernel memory 86

L

license inquiries 399
linking
 filesets 205
links to snapshots
 creating 277
 deleting 277
LookAt message retrieval tool xii
lost+found directory 20

M

managing
 a GPFS cluster 5
 GPFS quotas 41
maxblocksize attribute
 changing 95
maxFilesToCache attribute
 changing 95
maximum number of files
 changing 109
 displaying 219
maxMBpS attribute
 changing 95
maxStatCache attribute
 changing 95
message retrieval tool, LookAt xii
metadata 87
metadata replica 110

metadata replication
 changing 23
mmaddddisk 32, 66, 392
mmaddnode 6, 70
mmapplypolicy 73
mmauth 78
mmbackup 28, 83
mmchattr 23, 86
mmchcluster 8, 89
mmchconfig 9, 58, 59, 92
mmchdisk 36, 37, 100
mmcheckquota 41, 43, 104
mmchfileset 107
mmchfs 22, 45, 59, 109
mmchmgr 114
mmchnode 116
mmchnsd 120
mmchpolicy 122
mmcrcluster 5, 124
mmcrfileset 128
mmcrfs 17, 45, 53, 59, 130
mmcrnsd 32, 137, 271, 391, 392
mmcrsnapshot 29, 142
mmcrvsd 145
mmdefedquota 150
mmdefquotaoff 152
mmdefquotaon 154
mmdefragfs 26, 27, 157
mmdelacl 52, 53, 56, 160
mmdeldisk 162, 392
mmdelfileset 165
mmdelfs 19, 168
mmdelnod 7, 170
mmdelnod 172
mmdelsnapshot 174
mmdf 25, 176
mmeditacl 52, 53, 55, 56, 179
mmedquota 43, 182
mmexportfs 185
MMFS_FSSTRUCT 187
MMFS_SYSTEM_UNMOUNT 187
mmfsck 187
mmfsctl 192
mmgetacl 51, 55, 56, 196
mmgetstate 199
mmimportfs 202
mmlinkfileset 205
mmlsattr 23, 207
mmlscluster 5, 209
mmlsconfig 211
mmlsdisk 36, 213
mmlsfileset 216
mmlsfs 21, 59, 219
mmlsmgr 13, 223
mmlsmount 19, 225
mmlsnsd 31, 227
mmlspolicy 230
mmlsquota 44, 232
mmlssnapshot 235, 328
mmmount 17, 237
mmnsddiscover 239

- mmpmon 241
- mmpuac 50, 51, 53, 55, 56, 246
- mmquotaoff 45, 249
- mmquotaon 45, 251
- mmremotecluster 253
- mmremotefs 256
- mmrepquota 46, 259
- mmrestorefs 262
- mmrestripefile 265
- mmrestripefs 24, 268
 - completion time 13
- mmrpldisk 271
- mmshutdown 14, 275
- mmsnapdir 277, 308
- mmstartup 14, 280
- mmtracectl 282
- mmumount 18, 19, 285
- mmunlinkfileset 288
- modifying file system attributes 22
- mount point directory 131
- mounting
 - file systems 17
- mounting a file system 109
 - an NFS exported file system 57
- mtime 132, 351, 364, 393

N

- Network File System (NFS)
 - cache usage 59
 - exporting a GPFS file system 57
 - interoperability with GPFS 57
 - synchronous writes 59
 - unmounting a file system 59
- Network Shared Disks (NSDs) 391
 - automatic creation 202
 - changing configuration attributes 38, 120
 - creating 137
 - deleting 172
 - displaying 227
- NFS V4 49, 109, 131
- NFS V4 ACL 110, 133, 160, 179, 180, 196, 197, 246
 - GPFS exceptions 394
 - special names 394
- NFS V4 protocol
 - GPFS exceptions 394
- NIS automount 60
- node descriptor 70, 125
- node designation 70, 125
- node quorum 12
- node quorum with tiebreaker 8, 12
- nodes
 - adding to a cluster 70
 - adding to a GPFS cluster 6
 - assigned as file system manager 13
 - deleting from a cluster 170
 - which have file systems mounted 19
- notices 399
- NSD
 - path 239
- NSD fallback 39

- NSD server 13, 39, 67, 170, 202
- NSD server list
 - changing 38, 120
- NSD server node
 - choosing 137
- NSD server nodes
 - changing 38, 120
- NSD volume ID 137, 172
- nsdServerWaitTimeForMount
 - changing 95
- nsdServerWaitTimeWindowOnMount
 - changing 95

O

- operating systems
 - AIX, Linux, and Windows xi
- optimizing file access 61
- options
 - always 18
 - asfound 18
 - asneeded 18
 - atime 18
 - mount 17
 - mtime 18
 - never 18
 - noatime 18
 - nomtime 18
 - nosyncnfs 18
 - syncnfs 18
 - useNSDserver 18
- orphaned files 20

P

- pagepool attribute
 - changing 96
- partitioning file blocks 371
- patent information 399
- peer recovery cluster 192
- Peer-to-Peer Remote Copy (PPRC) 192
- performance
 - communicating file access patterns 61
 - monitoring 241
- persistent reserve
 - failureDetectionTime attribute 95
- Persistent Reserve
 - disabling 40
 - enabling 40
- policy
 - applying 73
- PR 40
- prefetchThreads attribute
 - changing 96
- primary GPFS cluster configuration server 125
- principles
 - common to GPFS commands 2
- problem determination information
 - placement of 93
- public/private key pair 78

Q

- querying
 - disk fragmentation 26
 - file system fragmentation 26
 - replication 23
 - space 25
- quorum 193
- quorum node 124, 170, 199
- quota files
 - replacing 47, 104
- quota information 360
- quotas
 - activating 251
 - activating limit checking 45
 - changing 43, 182, 357
 - checking 43, 104
 - creating reports 259
 - deactivating 249
 - deactivating quota limit checking 45
 - default values 42
 - disabling 42
 - displaying 44, 232
 - enabling 41
 - establishing 43
 - fileset 41
 - group 41
 - reports, creating 46
 - setting 182
 - user 41

R

- RAID stripe size 130
- rcp 68, 71, 75, 84, 90, 98, 102, 107, 112, 114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166, 168, 171, 173, 175, 177, 185, 189, 194, 199, 203, 205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266, 269, 273, 276, 278, 280, 289
- read file permission 49
- rebalancing a file 265
- rebalancing a file system 268
- refresh NSD server
 - mmnsddiscover 239
- remote copy command
 - changing 89
 - choosing 124
- remote shell command
 - changing 89
 - choosing 124
- repairing a file system 20
- replacing disks 34
- replicated cluster 392
- replication 110, 133
 - changing 23
 - querying 23, 207
- replication attributes
 - changing 86
- replication factor 86
- requirements
 - administering GPFS 1

- restoring NSD path
 - mmnsddiscover 239
- restripe *see* rebalance 409
- restriping a file 265
- restriping a file system 24, 268
- root authority 1
- rsh 68, 71, 75, 84, 90, 98, 102, 107, 112, 114, 121, 126, 128, 135, 139, 143, 148, 158, 163, 166, 168, 171, 173, 175, 177, 185, 189, 194, 199, 203, 205, 209, 211, 214, 217, 220, 223, 228, 235, 263, 266, 269, 273, 276, 278, 280, 289

S

- secondary GPFS cluster configuration server 126
- server node
 - restoring NSD path 239
- server node, NSD
 - choosing 137
- setting access control lists 50
- sidAutoMapRangeLength attribute
 - changing 96
- sidAutoMapRangeStart attribute
 - changing 96
- snapshot directory 321
- snapshot handle 308, 312, 313, 314, 316, 318, 320, 323
 - free 307
- snapshot ID 308, 309, 313, 318
 - comparing 295
 - internal 328
- snapshot name 314, 323
- snapshots
 - creating 142
 - deleting 174
 - directory 142
 - displaying 235
 - listing 235
 - maximum number 142
 - partial 142
 - restoring a file system 262
- sparse file 345
- specifying
 - subnets attribute 96
- standards, exceptions to 393
- starting GPFS 14, 280
 - before starting 14
- status
 - disk 36
- stopping GPFS 14
- storage
 - pre-allocating 353
- storage pools
 - ID 335
 - name 335
- strict 110, 133
- Stripe Group Manager *see* File System Manager 409
- structures
 - gpfs_acl_t 293
 - gpfs_direntx_t 297
 - gpfs_fssnap_handle_t 308

structures (continued)

- gpfs_fssnap_id_t 309
- gpfs_iattr_t 326
- gpfs_ifile_t 330, 331
- gpfs_iscan_t 347
- gpfs_opaque_acl_t 350
- gpfs_quotaInfo_t 360
- gpfsAccessRange_t 366
- gpfsCancelHints_t 367
- gpfsClearFileCache_t 368
- gpfsDataShipMap_t 369
- gpfsDataShipStart_t 371
- gpfsDataShipStop_t 374
- gpfsFcntlHeader_t 375
- gpfsFreeRange_t 376
- gpfsGetFilesetName_t 377
- gpfsGetReplication_t 378
- gpfsGetSnapshotName_t 380
- gpfsGetStoragePool_t 381
- gpfsMultipleAccessRange_t 382
- gpfsRestripeData_t 384
- gpfsSetReplication_t 385
- gpfsSetStoragePool_t 387

subnets attribute

- specifying 96

subroutines

- gpfs_close_inodescan() 294
- gpfs_cmp_fssnapid() 295
- gpfs_fcntl() 61, 298
- gpfs_fgetattrs() 301
- gpfs_fputattrs() 303
- gpfs_fputattrswithpathname() 305
- gpfs_free_fssnaphandle() 307
- gpfs_fstat() 310
- gpfs_get_fsname_from_fssnaphandle() 312
- gpfs_get_fssnaphandle_by_fssnapid() 313
- gpfs_get_fssnaphandle_by_name() 314
- gpfs_get_fssnaphandle_by_path() 316
- gpfs_get_fssnapid_from_fssnaphandle() 318
- gpfs_get_pathname_from_fssnaphandle() 320
- gpfs_get_snapdirname() 321
- gpfs_get_snapname_from_fssnaphandle() 323
- gpfs_getacl() 324
- gpfs_iclose 29
- gpfs_iclose() 329
- gpfs_igetattrs() 331
- gpfs_igetfilessetName() 333
- gpfs_igetstoragepool() 335
- gpfs_iopen 29
- gpfs_iopen() 337
- gpfs_iread 29
- gpfs_iread() 339
- gpfs_ireaddir 29
- gpfs_ireaddir() 341
- gpfs_ireadlink() 343
- gpfs_ireadx() 345
- gpfs_next_inode 29
- gpfs_next_inode() 348
- gpfs_open_inodescan 29
- gpfs_open_inodescan() 351
- gpfs_prealloc() 353

subroutines (continued)

- gpfs_putacl() 355
- gpfs_quotactl() 42, 357
- gpfs_seek_inode() 362
- gpfs_stat() 364

supported operating systems

- AIX, Linux, and Windows xi

symbolic link

- reading 343

- syncFSconfig 192

T

- tiebreaker disk 172

tiebreakerDisks

- changing 97

- time it will take to detect that a node has failed

- failureDetectionTime attribute 95

- timeout period 275

Tivoli Storage Manager (TSM)

- documentation 28

- setup requirements 28

- using the mmbackup command 83

- version 28

traceFileSize

- changing 283

traceRecycle

- changing 282

trademarks 400

- traditional ACL 110, 133, 179, 180, 196, 197

U

- UID domain 126

uidDomain attribute

- changing 97

unlinking

- filesets 288

- unmounting a file system 18

- NFS exported 59

- on multiple nodes 19

unmountOnDiskFail attribute

- changing 97

useNSDserver

- values 18

usePersistentReserve

- changing 97

user exits 389

- mmsdrbackup 390

- nsdddevices 391

- syncfsconfig 392

- user quota 151, 152, 154, 183, 232, 251, 259

- user space buffer 86

- using a clustered NFS subsystem 60

V

verbsPorts

- changing 97

verbsRdma

- changing 98

virtual shared disk server 146
virtual shared disks 137
 creating 145

W

worker1Threads attribute
 changing 98
write file permission 49

Reader's comments - We'd like to hear from you

**General Parallel File System
Administration and Programming Reference
Version 3 Release 2.1**

Publication No. SA23-2221-02

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 58HA, Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5724-N94
5765-G66

SA23-2221-02

