TWIN PEAKS SOFTWARE

MIRROR FILE SYSTEM 2.0
System Administration Guide
Linux x86 Platform Edition

Twin Peaks Software Inc.
46732 Fremont Blvd.
Fremont, CA 94538 U.S.A.
Tel: (510) 438-0536 Fax: (717) 427-3470
www.TwinPeakSoft.com

Revision L, May 2009

Copyright and Disclaimer

Contents

Chapter 1 What is the Mirror File System?

Twin Peaks Software Mirror File System™ (MFS™) is a file system for Linux operating environment on Intel and AMD x86 platforms. The Mirror File System enables a Linux system to mirror (replicate) the contents of its local file system, such as the Linux file system EXT3, across the network to another EXT3 file system on remote host in real time via the NFS protocol. (See Figure 1)

The Mirror File System is a virtual, or stackable, file system that has no physical disk or storage directly underneath it. A stackable file system sits on top of the other file systems and controls the file systems below it. For release 2.0, the MFS can manage two file systems, one EXT3 and one NFS (client side mounted NFS). When an application initiates an I/O request, the MFS receives it first, then forwards the I/O to the file systems underneath it.

The Mirror File System can also be exported or shared to clients on the network via the NFS protocol.

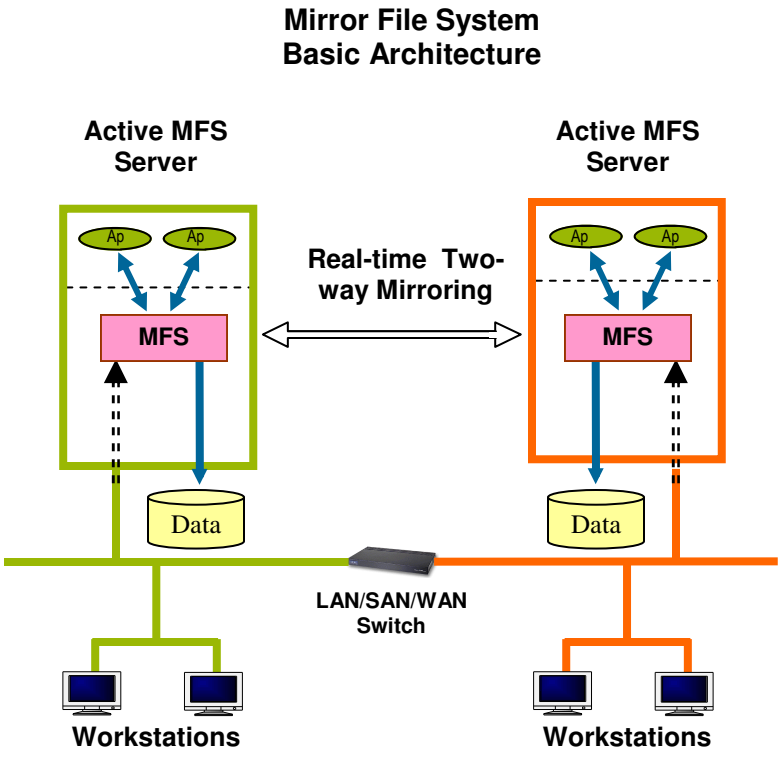
Mirror File System
Basic Architecture
Active MFS
Server
Active MFS
Server
Ap
Ap
Ap
Ap
Real-time Two-
way Mirroring
MFS
MFS
Data
Data
LAN/SAN/WAN
Switch
Workstations
Workstations

Figure 1

Chapter 2 **How Does the Mirror File System Work?**

In order for a file system to be activated from the disk or storage, the file system needs to be mounted on a mount point. The mount point is normally the pathname of a directory of an already mounted file system, such as the root file system. Once a file system is mounted, an application can access the contents of the file system under the mount point through the mount point directory. (See Figure 2)

The MFS has a unique mounting protocol that offers several improvements over other file system mounting mechanisms:

- When a directory of a remote NFS file system specified as `hostname:/pathname` is mounted on a mount point of a local EXT3 file system through the MFS mount command, the content of the local EXT3 under the mount point becomes the local copy of MFS, and the content of the newly-mounted NFS becomes the remote copy of MFS. The MFS 2.0 release supports one local copy (EXT3) and one remote copy (NFS) under a single mount point.
- After the MFS mount, the contents of the mount point are controlled by the MFS instead of its original EXT3 file system. The newly-mounted NFS is also under MFS control. (The MFS sits on top of EXT3 and NFS on the mount point.)
- The contents of the local copy (EXT3) may not be identical to the contents of the remote copy (NFS). The MFS File System Consistency Check utility, mfsck, checks and synchronizes the contents of both the local copy (EXT3) and the remote copy (NFS), thus ensuring that their contents are identical. (See the `mfsck` man page for more details.)
- After `mfsck` is done, there are two identical copies of the same contents under mount point, one on the local EXT3 file system, the other on the NFS file system on the remote NFS server.
- When an application accesses the contents on mount point, the MFS receives the operation first. It forwards READ operations to the local EXT3 file system to get the data and forwards WRITE operations to both the EXT3 and the NFS. A WRITE operation

thus updates the contents of both file systems under the mount point. (This is an operational definition of mirroring.)

- The application sees only one copy of the data. All MFS operations are transparent to the application.

Mirror File System
Server Basic Configuration

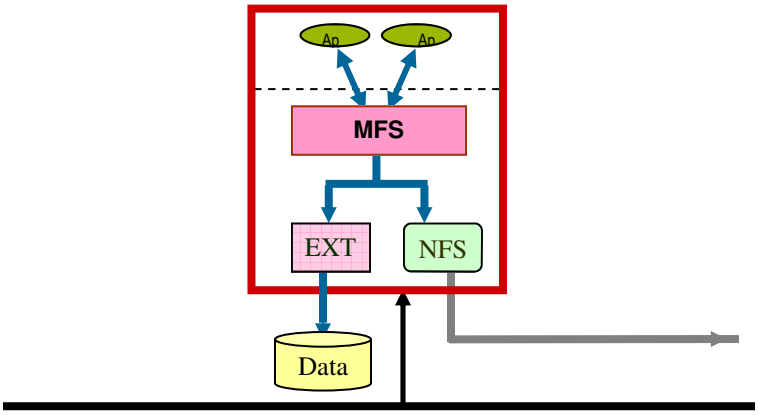

Figure 2

Chapter 3 **How to Configure MFS**

There are two basic ways to configure a Mirror File System:

- Active-Passive
- Daisy Chain

Active-Passive Configuration

In this configuration, a system with MFS mirrors a local EXT3 to a remote NFS. This setup uses one-way mirroring, from an active system to a passive system. (See Figure 3)

Configure a Passive System

To be configured as a passive system in an MFS implementation, a Linux or other Unix system that supports the NFS protocol needs to share (export) the contents of its EXT3 filesystem with the active remote MFS server. Sharing the contents of its filesystem with the active server enables the passive server to receive mirrored data from the active server. For information on sharing (exporting) file systems via the NFS protocol, see the man pages for the current operating system.

For example, passive MFS server Peak10, which receives the mirrored data from the active system, needs to become a NFS server and export the pathname of its EXT3 to the active system, Peak221, by adding the following command to the `/etc/exports` file.

```
/Peak10/user/home Peak221(rw,async,no_root_squash)
```

The `rw` option gives Peak221 read and write permission. The `async` option can improve the performance. Please review exportfs man page about option of `async` and `sync`. The `no_root_squash` option gives root access to Peak221. The MFS utilities `mfsck` (MFS file system consistency check) and `msync` (MFS file system synchronizer) need root permission to work correctly. See the `mfsck` and `msync` man pages for more details. After adding the new exported file system entry, the system administrator needs to run the `/etc/init.d/nfs restart` command to start NFS daemon processess and run the `/usr/sbin/exportfs -va` command to export the

/Peak10/user/home ddirectory to Peak221. Please review man pages of exports, exportfs and nfs for more details.

Configure an Active System

For an active MFS system Peak221 be able to mirror the contents under its /Peak221/export/home directory to the /Peak10/user/home directory on Peak10 system. The system administrator needs to run the mfs mount command after the passive system Peak10 finishes its setup.

Before the mount operation, all applications, daemons that currently access the file, directory under the mount point (/Peak221/export/hom in the example) have to exit, so all files, directories are in quiescent state. If the files, directories are not quiescent, then existing appplications still access the files, directories through EXT3 file system and the appllications, daemons started after MFS mount operation will access the files , directories through the newly mounted MFS, this will cause inconsistency between the MFS and the underneath EXT3 file systems. The inconsistency may trigger a system panic or errors during file system umount operation.

```
Peak221# mount –t mfs Peak10:/Peak10/user/home
/Peak221/export/home
```

System administrator can run the MFS mount command manually or put the MFs mount command in `start` section of /etc/rc.d/rc3.d/S25nfsfs script so the MFS mount is executed during he system boot up process. One can also add a MFS mount entry in /etc/fstab file to faciliate the MFS mount operation.

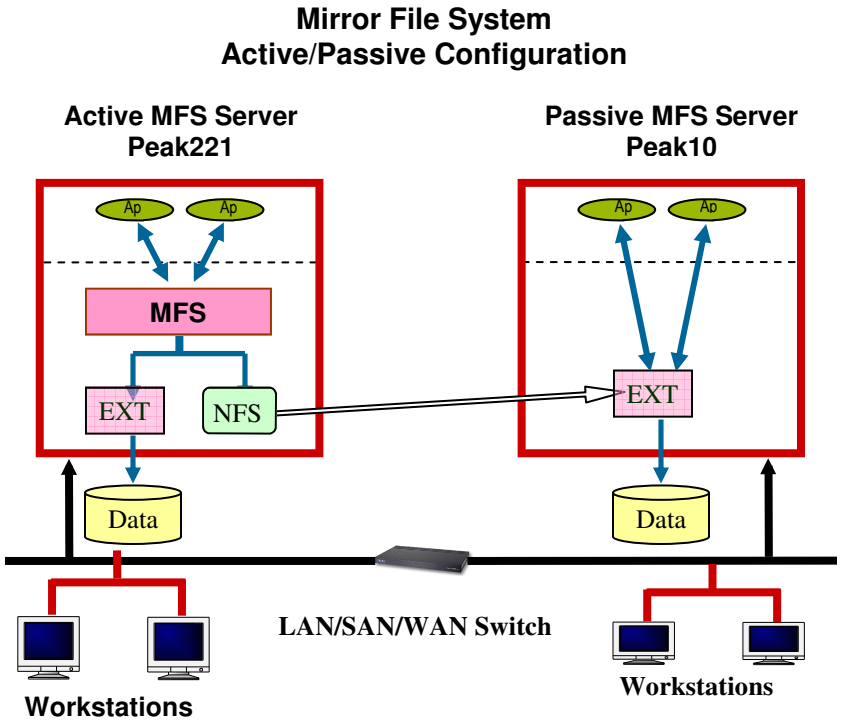

Figure 3

To look at the mounted file system after the mfs mount command finishes, type:

```
Peak221# df -T | grep '/Peak221/export/home'
     mfs   -- -- --   16%    /Peak221/export/home
```

The mount point /Peak221/export/home is now an MFS mount point.

```
Peak221# mount -v | grep '/Peak221/export/home'

Peak10:/Peak10/user/home on /Peak221/export/home type mfs
(rw,soft,proto=tcp,addr=192.168.1.10)
```

The directory Peak10:/Peak10/user/home is mounted on /peak221/export/home, an MFS mount point.

Once the mount succeeds, Peak221 becomes an active MFS server. The contents in Peak10:/Peak10/user/home and /Peak221/export/home may not be identical. To make sure of these two tree structures of two file systems under MFS are containing identical content, the system admin **MUST** run the mfsck command to sync them up.

```
Peak221# mfsck /Peak221/export/home
```

The mfsck will fix any inconsistency between Peak10:/Peak10/user/home and /Peak221/export/home. Depending on the size of /Peak221/export/home directory, the mfsck may takes time to complete. In such case, the system admin can use command like rsync to shorten the mfsck time as show in Appendix B. When the mfsck command is completed, the application can do READ and WRITE operations on files and directories under the /Peak221/export/home directory on Peak221.

For the update-related operations create, write, and mkdir, the update goes to the /Peak221/export/home directory of Peak221 and the /Peak10/user/home directory of Peak10 in real time. Thus, the two directories on two separate systems contain identical information, updated in real time, all the time, until the setup is broken by the mfsumount command.

Because Peak10 is a passive MFS system, any updates on /Peak10/user/home in Peak10 system are stored only on Peak10 and are not mirrored to the /Peak221/export/home directory on Peak221.

It is recommended that the `/Peak10/user/home` is exported to be read only to all systems except the active MFS server Peak221.

Unconfigure an Active Configuration

To stop the active system Peak221 from mirroring, run the mfs umount command:

```
Peak221# /sbin/mfsumount /Peak221/export/home

Or

Peak221# /bin/umount /Peak221/export/hom
```

Note: The mfsumount command is specifically designed to unmount the MFS file system and hence unconfigure an active MFS server that runs on the older Linux system. The conventional /bin/umount command

can also unconfigure an active MFS system on the newer Linux system. Please see mfsumount man page for more details.

Unconfigure a Passive Configuration

To stop the passive system Peak10 from receiving mirrored data, run the exportfs command on Peak10:

```
Peak10# exportfs –u *:/Peak10/user/home
```

Daisy-Chain Configuration

Three or more MFS systems can be daisy-chained together in active-passive mode. If a system administrator wants to add another passive MFS server to serve the increasing number of clients on the network, or for use as a backup server, then a third MFS server can be added and connected to a running active-passive configuration. (See Figure 4)

To make the new system, Peak60, a passive MFS server, it needs to add the following share command entry to `/etc/exports` and issue `exportfs` commands as described in previous section to export EXT3 directory `/var/backup` to Peak10

```
/var/backup     Peak10(rw,async,no_root_squash)
```

Peak10, a passive MFS server, now becomes an active MFS server to Peak60 and but remains a passive server to Peak221. The following commands need to be run from Peak10:

1. unexport its EXT3 directory to Peak221.

```
Peak10# exportfs –u *:/Peak10/user/home
```

2. MFS mount Peak60:/var/backup on /Peak10/user/home

```
Peak10#  mount –t mfs Peak60:/var/backup /Peak10/user/home
```

3. After the mount completes, export its newly mounted MFS directory /Peak10/user/home to Peak221.

```
Peak10# exportfs -o async,rw,no_root_squash,fsid=2000
/Peak10/user/home
```

Because the MFS is a virtual file system, and is not mounted on a block device, the fsid is required to export the MFS to the active MFS server or other clients.

When all commands succeed, the final configuration looks like Figure 5.

The first system, Peak221 is an active MFS system to the second system Peak10; the second system Peak10 is a passive system to the first system Peak221 but an active system to the third system, Peak60. The third system, Peak60, is a passive system to the second system Peak10. When the contents of the MFS in Peak221 are updated, the updates are propagated to the second system, Peak10, then to the third system, Peak60.

The Peak60 system is at the end of a daisy chain. It can be an active system to another system, which must be chained after it, but it cannot be an active system to Peak221, which is the head of this daisy chain, otherwise an infinite loop would be created.

All MFS configurations, active-passive and daisy-chain, can be performed either during the initial setup, after the servers are booted up, or on the fly, when one MFS server is already running and serving clients. The MFS File System Consistency Check (the `mfsck` command) can also be run either during the initial setup or on the fly. In either case, it detects and fixes any inconsistency between two file systems under MFS and makes sure the two file systems are identical and consistent.

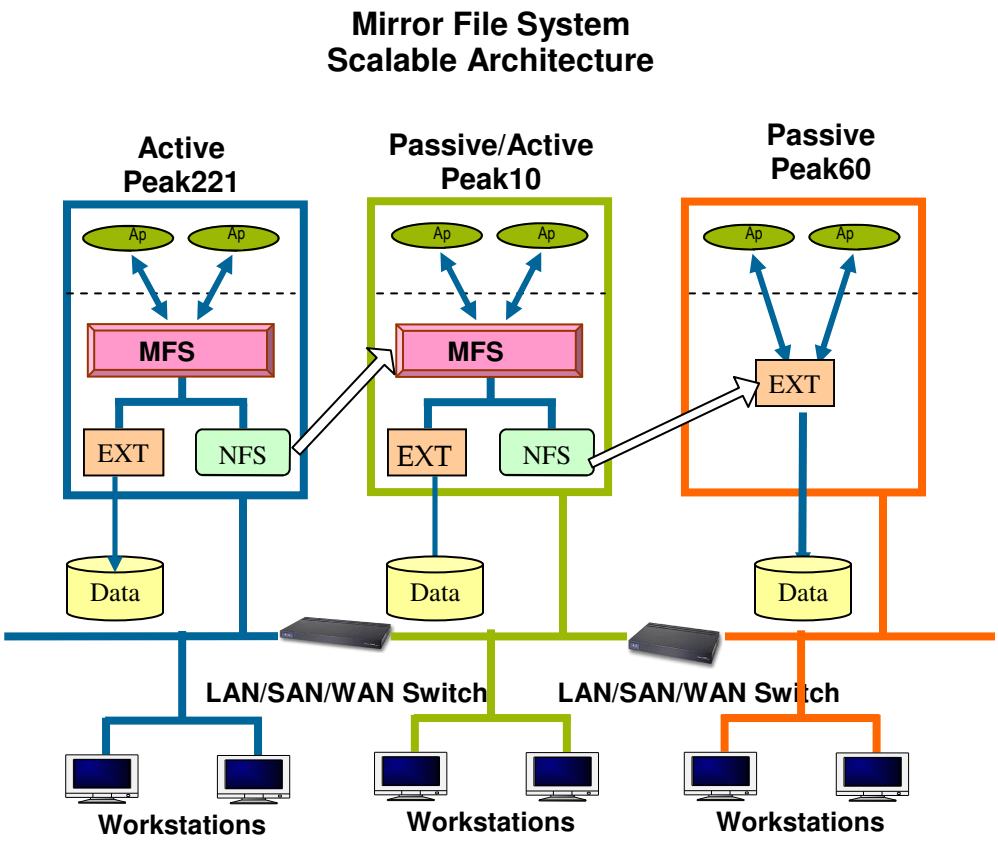
Mirror File System
Scalable Architecture
Active
Peak221
Passive/Active
Peak10
Passive
Peak60
Ap
Ap
Ap
Ap
Ap
Ap
MFS
MFS
EXT
EXT
NFS
EXT
NFS
Data
Data
Data
LAN/SAN/WAN Switch
LAN/SAN/WAN Switch
Workstations
Workstations
Workstations

Figure 4

Chapter 4 Failover and `msync` Operations

Both active and passive MFS systems may encounter the following situations during operation.

The Active Server is Down

When the active system is equipped with the open source Heartbeat package from http://www.linux-ha.org, the active-passive system can be set up as a robust, failsafe clustering system. When the active MFS server goes down or taken out for service, the passive server can immediately take over the IP address of the clustering system, and all of its operations, and become the primary server. Applications and clients continue to get the most up-to-date information without knowing that a failover operation has happened between active and passive servers. When the active server comes back on line, a failback operation can restore the active server to its role as the primary server.

Please see Appendix A for an example.

The Passive Server is Down

If the passive MFS server crashes or shuts down, the active MFS server continues normal operation, and updates are sent only to file systems and storage associated with the active MFS server. The MFS on the active server logs the names of files that cannot be replicated to the passive server. When the passive server is back online again, the msync daemon running on the active server reads the log and re-transmits the missing files to the passive server.

The `msync` daemon uses a log with a capacity for 262,144 (256K) file/directory names, to allow the passive server to be taken out for service for very extensive period if necessary. When it is brought back on line, all missing file/directory operations are re-syncd with the running active MFS server.

The Network is Down

A temporary network glitch or outage can cause two MFS servers working as partners to become disconnected for a short period of time, and one server may be out of sync with the other one as a result. The active server treats this situation as though the passive server is down, as described above. When the network outage is over, the `msync` daemon checks its log to find out what files/directories are out of sync between the two servers and starts the re-sync process to make sure both servers contain consistent and up-to-date information. This re-sync takes place automatically, without system administrator intervention. The system adminstrator needs only to start the `msync` daemon after the MFS mount comand.

The time for a MFS server to confirm that its remote MFS partner is down is 120 seconds (default) or 20 seconds if using TCP or UDP prootocol in the MFS mount command.

Chapter 5 **Mount Options**

The MFS `mount` command supports all generic options listed in the `mount (1)` man page. The following specific options are supported:

1. All other specific options supported are the same as those supported by the NFS `mount` command (See nfs(1) man page). The only exception is the hard mount option. The MFS does not support NFS client hard mount.

2. The MFS uses the soft mount option to mount the NFS. The soft option allows MFS to break away from the remote passive server without continually retrying a failed operation if there is a network outage or if the remote passive server is down.

Chapter 6 **Sharing MFS with Clients**

MFS uses the NFS protocol to export the MFS to its clients on the network.

The system administrator can run the exportfs command after setting up `/Peak221/export/home` as an active MFS directory.

```
Peak221#  exportfs –o rw,fsid=2001  /Peak221/export/home
```

The `exportfs` command exports s `/Peak221/export/home` MFS directories to clients. As described above, the fsid is required for a virtual file system to be exported. The client system can do an NFS client mount:

```
Client# mount Peak221:/Peak221/export/home /Peak_up/home
```

The client can NFS mount the exported MFS with options described in nfs man page. The update made on /Peak221/export/home from Client system is sent to active MFS server Peak221.

Once received the update from Client , MFS on Peak221 forwards update to both local EXT3 file system on Peak221 and the EXT3 file system on remote passive server Peak10. In case Peak221 is not available as a result of network outage or maintenance shutdown, Client can switch to /Peak10/user/home directory of Peak10 and continue the operation without interruption.

Chapter 7 **MFS Utilities**

MFS File System Consistency Check utility - mfsck

`mfsck` checks and repairs inconsistencies between two file systems under the MFS mount point. It can be run at initial active/passive setup or on the fly when the MFS is fully operational, and is a good tool for auditing and fixing inconsistencies between local and remote copies of the MFS. The system administrator can run `mfsck` on any MFS directories whenever a consistency is in doubt.

```
Peak221# /sbin/mfsck /Peak221/export/home/johndoe
```

MFS File System Synchronizer Daemon - msync

The `msync` daemon guards against any inconsistency while the MFS is running. Inconsistency can occur as the result of network glitches, outages, the remote passive server temporarily being taken down, or other planned or unexpected events.

The `msync` daemon should start right after the MFS is mounted and stay operational until the MFS is unmounted. When a problem occurs while an active MFS server is sending updates to another remote passive server, the active server starts to log the files/directories that did not get sent over to the remote MFS server. When the problem is resolved and the remote passive server is reconnected to the active MFS server, the `msync` daemon sends those files/directories to the remote passive server. The resend operation of `msync` happens automatically, without system administrator intervention.

```
Peak221# msync /Peak221/export/home
```

Chapter 8 **MFS Man Pages**

As of the current release, the Mirror File System uses four commands that can be accessed with the **man** command:

NAME

mount_mfs - mount a MFS resources

SYNOPSIS

```
/bin/mount -t mfs [ generic_options ]
      [ -o specific_options ] [ -O ] resource
/bin/mount -t mfs [ generic_options ]
      [ -o specific_options ] [ -O ] mount_point
/bin/mount -F mfs [ generic_options ]
      [ -o specific_options ] [ -O ] resource mount_point
```

DESCRIPTION

The MFS mount utility attaches a named NFS resource to the file system hierarchy at the pathname location mount_point, which must be an already existing EXT3 directory. The contents under mount_point prior to the mount operation becomes the local copy of MFS, the contents in named NFS resource become remote copy of MFS. These two copies should be identical at all times under the MFS. If there is any inconsistency between these two copies, the MFS utility mfsck--MFS file system consistency check can be invoked to fix this issue.

When the application accesses the contents of MFS and the operation is a READ, MFS will only need to forward the operation to the local copy, the EXT3. If the operation is a WRITE, MFS will forward the operation to both the local copy (EXT3) and the remote copy (NFS). Hence, the mirroring is accomplished.

Before the mount operation, all applications, daemons that currently access the file, directory under the mount point have to exit, so all files, directories are in quiescent state. If the files, directories are not quiescent, then existing applications still access the files, directories through EXT3 file system and the appllications, daemons started after MFS mount operation will access the files, directories through the newly mounted MFS, this will cause inconsistency between the MFS and the underneath EXT3 file systems. The inconsistency may trigger a system panic or errors during file system umount operation.

If the resource is listed in the `/etc/fstab` file, the command line can specify either resource or mount_point, and mount will consult `/etc/fstab` for more information. If the -F option is omitted, mount takes the file system type from `/etc/fstab`.

If the resource is not listed in the `/etc/fstab` file, then the command line must specify both the resource and the mount_point.

A named NFS resource can have one of the following formats:

`host:pathname`
Where host is the name of the NFS server host, and pathname is the path name of the directory on the server being mounted. The path name is interpreted according to the server's path name parsing rules and is not necessarily slash-separated, though on most servers, this will be the case.

`nfs://host[:port]/pathname`
This is an NFS URL and follows the standard convention for NFS URLs as described in Internet RFC 2225 - NFS URL Scheme. See the discussion of URL's and the public option under NFS FILE SYSTEMS below for a more detailed discussion.

`nfs://host[:port]/pathnameresources`
A comma-separated list of host:pathname and/or See the discussion of Replicated file systems and detailed discussion. mount maintains a table of mounted file systems in `/etc/mnttab`, described in mnttab(4).

OPTIONS

See mount (1) for the list of supported generic_options.

-o specific_options

Since the MFS mount is mounting an NFS on a EXT3 mount_point, all specific_options supported by nfs (1) are supported by MFS, except the '-o hard' (hard mount) option. The MFS does not support NFS hard mount option.

EXAMPLES

Example 1: Mounting an MFS File System
```
example# mount -t mfs serv:/usr/src /usr/src
```

Example 3: Mounting an MFS File System over NFS with the UDP Transport. The default is TCP.
```
example# mount -t mfs -o proto=udp serv:/usr/src /usr/src
```

FILES

`/etc/fstab` file systems table

`/etc/mtab` table of mounted file systems

SEE ALSO

mount(2), mfsumount(1), mtab(4), mfsck(1), msync(1), nfs(1)

User commands **mfsumount(1)**

NAME

mfsumount - umount a MFS resources

SYNOPSIS

```
/sbin/mfsumount   mount_point
/bin/umount       mount_point
```

DESCRIPTION

A mounted MFS file system has two file systems under its mount point. The MFS umount utility unmounts these two file systems:

1. unmounts a MFS from the mount_point
2. unmounts a currently mounted NFS resource from the mount_point

Unlike the conventional system umount command /bin/umount, which only unmounts a single file system from a mount point, the /bin/mfsumount executes two file system umount.

Ideally, the Linux operating system should provides the same mechanism to both mount and umount commands, so that both mount and umount commands invoke file system-specific mount and umount comamnds. For example, /bin/mount.mfs is the MFS mount command that invoked by the /sbin/mount command with a -t mfs argument. Because the same mechanism is not provided in /sbin/umount command, however, it is advisable to use /sbin/mfsumount command to unmount a MFS file system from a mount_point.

Note: The newer version of Linux provides the mechanism to invoke /sbin/umount.mfs from /bin/umount command. Please verify this on your Linux system.

The table of currently mounted file systems can be found in /etc/mtab. Mounting a file system adds an entry to the mount table; a umount removes an entry from the table.

EXAMPLES

Example 1: Unmounting an MFS File System

```
example# /sbin/mfsumount /usr/src
   or
example# /bin/umount /usr/src
```

FILES

/etc/fstab file system table

/etc/mtab table of mounted file systems

SEE ALSO

mfsck(1), msync(1), mfsumount(1), fstab(5), mount(8), umount(8), nfs(5), exports(5),

User Commands **mfsck(1)**

NAME

mfsck - check and synchronize Mirror File Systems

SYNOPSIS

```
/sbin/mfsck [ -n ]  mfs_dir
/sbin/mfsck [-y] [ -f | b ] [ -i | v ] [ -t ] mfs_dir
```

DESCRIPTION

mfsck checks and repairs inconsistencies between two file systems, EXT3 and NFS, under MFS. If an inconsistency is detected, the default action for each correction is to make both the local (EXT3) and remote (NFS) file system under MFS identical. If the local file system (EXT3) has files/directories which are not in the remote file system (NFS), those files/directories will be copied to the remote file system (NFS), and vice versa. If a file/ directory exists in both file systems, but their contents are different, the default action is to copy the files/directories in the local file system (EXT3) to the remote file system (NFS). The user can choose to decide interactively how to resolve each inconsistency under MFS.

OPERAND

The mfs_dir is a directory of MFS. The user can choose to repair the inconsistency on a particular directory of MFS. The user can also run more than one mfsck to repair several directories concurrently for better performance. The MFS allows up to 256 mfsck to run concurrently.

OPTIONS

The following generic options are supported:

-n Check but do not repair. This option checks whether the MFS is inconsistent. If inconsistencies are detected, it will print out the inconsistent files/directories.

-f | -b Determine the synchronization directions. The -f option does forward synchronization, which synchronizes (copies) the local file system (EXT3) to the remote file system (NFS). The -b option does reverse synchronization by synchronizeing the remote file system(NFS) to local file system(EXT3).

-v Verbose mode. This option prints out the inconsistent files/directories.

-i Interactive mode. This option will wait for the user's response whenever an inconsistency is found. The user can choose forward or reverse synchronization, or do nothing. The user also has the option to exit interactive mode.

-t Thoroughly compare two files/directories from both file systems. The default is a quick check for inconsistency. This option will compare, byte by byte, of two files/directories to determine the inconsistency.

-y Non-interactive mode. This option will do the forward (-f) or backward (-b) operation without asking user to confirm.

EXAMPLES

Example 1: To check and fix inconsistencies in `/export/home/johndoe`

```
example# /sbin/mfsck /export/home/johndoe
```

Example 2: To check, byte by byte, and fix inconsistencies in `/export/home/johndoe`. When an inconsistency is detected, copy the local copy (EXT3) to the remote copy (NFS).

```
example# /sbin/mfsck -t /export/home/johndoe
```

Example 3: To check byte by byte and fix inconsistency in `/export/home/johndoe`. When an inconsistency is detected, copy the remote copy (NFS) to the local copy (EXT3).

```
example# /sbin/mfsck -tb /export/home/johndoe
```

EXIT STATUS

0 No inconsistencies in the MFS

>0 Inconsistencies exsist in the MFS

FILES

`/var/mfs/mfsck/nfstmp_xxxx`

mount point for the remote system during mfsck

SEE ALSO

msync(1), mount_mfs(1)

User Commands **msync(1)**

NAME

msync - synchronize Mirror File Systems after a remote file system failure

SYNOPSIS

```
/sbin/msync mfs_root_dir
```

DESCRIPTION

msync is a daemon used to synchronize an MFS when a remote file system (NFS) failure occurs. When the remote file system (NFS) under an MFS is temporarily unavailable, MFS will update only the local file system (EXT3), and log those updates to a log. When the remote file system (NFS) comes back on line again, the msync daemon will access the entries in the log and update the remote file system (NFS) accordingly to make both file systems consistent. The default size of the log is 262,144 (256K) file/directory pathname entry.

OPERAND

The mfs_root_dir is the mount_point of the MFS. The daemon is usually run after the MFS mount succeeds. It will stay operational until the MFS is unmounted.

EXAMPLES

Example 1: Run the msync after the MFS mount is successful

```
example# mount -t mfs Peak220:/export/home /export/home
   example# /sbin/msync /export/home
```

Example 2: To kill the msync daemon

```
example# ps -ef | grep msync
 root 400 1  0  Dec 08 pts/1 0:01 /sbin/msync /export/home
example# kill -9 400
```

EXIT STATUS

0 Daemon started successfully.
>0 Daemon failed to start.

FILES

`/var/mfs/msync/*`
system log files for the undone operations

SEE ALSO

Mfsck (1), mount_mfs (1)

Appendix A -- Failsafe Live Clustering System

Please review the web page on how to install, configure heartbeat package with MFS to set up a Failsafe Live Clustering System

http://www.TwinPeakSoft.com/App_email.pdf

Appendix B - Recommended steps to set up large directory for MFS replication

A. Passive system, Peak10

1. Edit /etc/exports file and add the following entry
   /Peak10/user/home Peak221(rw,async,no_root_squash)

2. Clean up /Peak10/user/home directory
   # rm -rf /Peak10/user/home/*

3. reboot Peak10 to make sure it is in stable state.

B. Active system, Peak221

1. rsync(or tar) /Peak221/export/home directory on Peak221 to /Peak10/user/home directory on Peak10
   # rsync -ar /Peak221/export/home/* Peak10:/Peak10/user/home

   rsync will sync up contents in /Peak221/export/home to Peak10:/Peak10/user/home. This will help mfsck to speed up the sync later. All applications that access /Peak221/export/home directory can continue to run during rsync process.

2. As soon as rsync is complete, turn off daemons and applications that access files, directories under /Peak221/export/home directory
   For example,
   # chkconfig smb off
   # chkconfig vsftpd off

3. reboot Peak221
   # sync; reboot

4. After Peak221 is up, mount MFS on /Peak221/user/home
   #mount -t mfs Peak10:/Peak10/user/home /Peak221/user/home

5. run mfsck on /Peak221/user/home
   # mfsck -fy /Peak221/user/home

6. Turn on and start daemon and applications
   # chkconfig smb on
   # chkconfig vsftpd on
   # /etc/init.d/smb start
   # /etc/init.d/vsftpd start

C. MFS is ready to go