The mstore Command Reference

Also available in postscript and pdf versions.

Andrew Hanushevsky & Bill Weeks
SCS, Stanford Linear Accelerator Center
24-Sep-03
mstore create

1 Introduction

The mstore command provides a management interface between online Network File Systems (NFS-mounted disk space) and a Mass Storage System for experimental data on the order of approximately 100MB to 10GB per file. The mstore command allows you to create and delete file entries, as well as retrieve and store Mass Storage System managed data in a uniform way. The following table lists the command functions that are available.

<table>
<thead>
<tr>
<th>mstore Function</th>
<th>mstore Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>create one or more files</td>
</tr>
<tr>
<td>delete</td>
<td>delete one or more files</td>
</tr>
<tr>
<td>get</td>
<td>retrieve data from the Mass Storage System</td>
</tr>
<tr>
<td>ls</td>
<td>list files</td>
</tr>
<tr>
<td>put</td>
<td>store data in a Mass Storage System</td>
</tr>
<tr>
<td>rename</td>
<td>change the name of a file</td>
</tr>
<tr>
<td>set</td>
<td>set file attributes</td>
</tr>
</tbody>
</table>

Data managed by mstore can only be accessed via NFS. Two types of NFS space are available:
- Group space mounted at /nfs/mstore/g/grpname
- User space mount at /nfs/mstore/u/username

Group space is made available for defined groups to store experimental data. An mstore administrator must create the grpname entry. Afterwards, users who are members of the Unix group whose name is identical to grpname can use the mstore command to manage data in the directory sub-tree under grpname.

To request the creation of group space, fill out and submit the form at http://www.slac.stanford.edu/comp/unix/mstore-req.html

User space is made available to individual users for archival purposes. Any user whose Unix login name is identical to username can use the mstore command to manage data in the directory sub-tree under username.

Access to mstore managed data is via NFS-mounted disks. Any number of NFS file systems may be employed. However, primary access is always made via the mount-point /nfs/mstore. From there, symbolic links may lead you to other
NFS-mounted file systems where the actual data has been placed. In general, this process is transparent as long as you always start at /nfs/mstore.

In the previous diagram, the mstore file system is mounted at /nfs/mstore. This represents the mstore name space. All mstore managed files can be found using this mount point. The actual data, however, does not necessarily reside at this mount point. As examples, the file /nfs/mstore/u/abh/mydata is actually a symbolic link to a file residing in a file system mounted at /nfs/pubdata01. Similarly, the file /nfs/mstore/g/glast/data is actually a symbolic link to a file residing in a file system mounted at /nfs/pubdata01.

There are additional restrictions when using mstore managed space. These are:

- You cannot directly modify the directory structure (i.e., create or delete files) at /nfs/mstore. Instead, you must use the mstore create and mstore delete commands to do so.
- Directory and file names in mstore managed space are limited to the set of characters including upper and lower-case letter, digits, and the characters "-@#/%^_+=".
- While you can use the Unix chmod command to modify permission bits and chown or chgrp to change ownership or group affiliation for files, the changes are merely temporary. These Unix command only change the online state of the file. The offline copy of the file is not changed. This means that should the file be purged and then later brought back online, the permission bits, ownership, and group affiliation revert to those established when the file was created. Use the mstore set command to permanently change file attributes.
- The world-writable permission, any combination of execute permissions, as well as extended permission bits (e.g., setuid) are not supported.
- While you can use the Unix ls command to list the contents of mstore managed space, that listing may be misleading. This is because it only lists the current contents of the mstore disk cache that does not include files in the Mass Storage System. Use the mstore ls command to display a comprehensive list of files in a directory.

The typical sequence of events using mstore managed space includes:

- Using the mstore create command to create a file entry.
- Writing data into the file using standard Unix commands and programs (e.g., cp, tar, etc.).
- Using the mstore put command to force the data to be written to the Mass Storage System and possibly purging it from the disk cache.

---

1 Symbolic links are one-way transparent. A pwd command does not relate the symbolic link to the final destination.
mstore create

- Using the **mstore get** command to retrieve data from the Mass Storage System and making it available for reading and writing.
- Using the newly retrieved data as you would any NFS accessible file.
- Using the **mstore delete** command to remove all copies of the data (i.e., online and offline copies).

While the previous sequence included using the **mstore put** command to force writing the data into the Mass Storage System, there is no particular reason to do so unless you want to manually free up online disk space. The system automatically tracks changes to your files and writes changed files back to the Mass Storage System before removing the files from the online disk cache. When a file no longer exists online, you must use the **mstore get** command to copy the file from the Mass Storage System back to online disk space.

Because online disk space represents only a small portion of space used by all **mstore** managed files; it follows that not all files remain online. Indeed, if a file is not used for a system-determined period of time it is deleted from the online disk cache. If the file was modified, it is automatically written back to the Mass Storage System prior to deletion.

Mstore space is meant to store experimental data on the order of approximately 100MB to 10GB per file. Storing small files into **mstore** space will adversely affect overall performance. If you need to store many small files, you should use the Unix tar command to create a large archive of all of the files and store that tar file. Files larger than 10GB, while possible, represents a substantial intrusion on available space that is shared by many users. Contact the **mstore** administrator if you need to store files greater than 10GB.
1.1 Security Considerations

While your data resides in NFS-accessible space, somewhat different security rules apply. These rules make it easier for you to share the space while minimizing the possibility of introducing viruses and Trojan horse programs into the system. These rules are:

- Only machines that are members of the “scs-farm” NIS group can use `mstore` managed space.
- Only users whose Unix uid is in the range between 100 and 20,000 can use `mstore` managed space.
- Any member of the Unix group matching `grpname` in `/nfs/mstore/g/grpname` may create, delete, get, and put files in any directory sub-tree under `/nfs/mstore/g/grpname`.
- Only the user whose login name matches `username` in `/nfs/mstore/u/username` may create, delete, get and put files in any directory sub-tree under `/nfs/mstore/u/username`.
- Any user may get a file in any directory sub-tree under `/nfs/mstore/u/username` if that user has read access to the file.
- Only owner read/write, group read/write, and world read permission bits are supported.
2 mstore create

\texttt{mstore create [ options ] path}

\texttt{options: \texttt{-f} \texttt{-g} group \texttt{-K} keeptime \texttt{-m} mode

\hspace{1.5cm} \texttt{-P} purgetime \texttt{-v}}

Function
Create an \texttt{mstore} managed file.

Parameters
\texttt{path} is the name the file is to have. Specify one absolute \texttt{mstore} path or a path relative to the current \texttt{mstore} directory.

Options
\texttt{-f} silently deletes all copies of the identically named file before attempting to create the specified file.

\texttt{-g group}
is either a group id (i.e., gid) or a group name to be assigned to the file. You must be a member of the specified group. The default is to use your current effective group id. The \texttt{-g} option is ignored for group space files since those files are always assigned to the space group id.

\texttt{-K keeptime}
specifies the length of time the file is to be kept online. If you specify the word \texttt{forever}, the file is always kept online. You may also specify a date, as \texttt{mm/dd/[yy]yy} or \texttt{dd-mon-[yy]yy}, or a time, as \texttt{hh:mm:ss} until which the file is to be kept online. Finally, you may specify a duration, as \texttt{nnnn\[s | m | h | d]}, for the number of seconds, minutes, hours, or days (the default) that the file must be kept online. The default is to keep the file online until it is not used for 24 hours. You may be restricted to certain \texttt{keeptime} values.

\texttt{-m mode}
is either an absolute octal mode or a symbolic mode indicating the permission bits that are to be set for the file. See the notes for information about valid permissions. By default, the file is created with mode of 0644 masked with your current \texttt{umask} value.

\texttt{-P purgetime}
mstore create

specifies the length of time the file must be inactive before it can be removed from online space (i.e., purged). Specify a duration, as \texttt{nnmm[s|m|h|d]}, for the number of seconds, minutes, hours, or days (the default) that the file must be unused before it is purged. The default is to not purge the file unless it has not been used for 24 hours. You may be restricted to certain \textit{purgetime} values.

\textbf{\texttt{-v}} produces additional output during execution.

\textbf{Success}

The command ends with a status code of 0. A zero length file of \textit{path} is created.

\textbf{Failure}

The command ends with a non-zero status code.

\textbf{Notes}

1) Only one file may be created at a time.
2) Path names must begin with \texttt{/nfs/mstore/g/grpname} (group space) or \texttt{/nfs/mstore/u/username} (user space).
3) Characters in \textit{path} must be alpha-numeric or characters from the set of `~!@#$%^_+-=`.
4) For group space, the group space entry (\textit{grpname}) must already exist and you must be a member of the group. Mail the Unix administrator (\texttt{unix-admin}) to request the creation of a new group space entry, if necessary.
5) For user space, the \textit{username} portion of the path must match your username. If the path entry does not exist, it will be automatically created.
6) All intervening path components are automatically created for you, should they not exist. You cannot create directory entries either using NFS or through the \texttt{mstore} command.
7) Be careful when using the \texttt{-f} option. The \texttt{mstore} command first deletes all instances of the named file before it attempts the new creation. This means that you will loose the previous copy of the file.
8) It is possible that after creating a file, the file will not be visible via NFS. While the mstore command takes great care to minimize these occurrences, they may rarely occur. Should you encounter such an event, please report it to the Unix administrator.
9) Files can be created as dual copy files (i.e., a file will be stored twice, each copy on a different tape). However, you must request the mstore administrator to designate which mstore directories will be eligible for dual copies of files stored within them.
10) Absolute modes are composed of three octal digits. The first digit specifies the owner’s mode as 6xx (rw, the default), 4xx (r), or 2xx (w). The second digit specifies the group’s mode as x6x (rw), x4x (r, the default), x2x (w), or x0x (none). The third digit specifies the world’s mode as xx4 (r, the default) or xx0 (none). Extended permissions, execute permissions, and world write permissions are not respected. Additionally, the current \texttt{umask} value does not affect the resulting mode.

11) Symbolic modes follow the same rules as those defined for the Unix \texttt{chmod} command. Unlike absolute modes, the current \texttt{umask} value does affect the resulting mode. Otherwise, all other restrictions apply to the mode as described above.

12) In order to prevent Trojan horse programs from propagating throughout the system, \texttt{mstore} does not honor extended (e.g., setuid), execute, and world-write permission bits. While these permissions may be set on a temporary basis using the Unix \texttt{chmod} command, they are turned off when the file is brought back online using the “\texttt{mstore get}.”

13) Use the \texttt{mstore set} command to change a file’s group affiliation and permission bits. However, group affiliation may only be set for files in user space.

14) When a forced creation results in a file deletion, the deleted file must
   a. be owned by you or
   b. be assigned to a group in which you are a member and have the group write bit set (0020).
3 mstore delete

mstore delete [ options ] path [...]

options: -f  -r  -v

Function
Delete mstore managed files.

Parameters
path is the name of the directory or file to be deleted. Specify at least one absolute mstore path or a path relative to the current mstore directory.

Options
-f  does not prompt you to confirm the deletion of each specified path.
-r  should path be a directory, recursively deletes all sub-entries in the directory as well as the directory itself.
-v  produces additional output during execution.

Success
The command ends with a status code of 0.

Failure
The command ends with a non-zero status code.

Notes
1) Generally, the path should not be a glob. If you do specify a glob, the shell will expand the glob to a list of matching files. This list usually will not be representative of all stored files since some may not be online. Using globs may lead to unpredictable deletions.
2) Each path must start with
   /nfs/mstore/g/grpname or /nfs/mstore/u/username
   and your uid must either be a member of a group matching grpname or
   must map to username.
mstore delete

3) Each deleted file must
   a. be owned by you and have owner or group write bit (0220),
   b. be owned by you and –f is specified, or
   c. be assigned to a group in which you are a member and have the
      group write bit set (0020).
4 mstore get

```
mstore get [ options ] path [ ... ]
```

**options:** `-b -K keeptime -q -P purgetime -v`

**Function**
Retrieve mstore managed files for reading or writing.

**Parameters**
`path` is the name of the file to be brought online. Specify at least one absolute mstore path or a path relative to the current mstore directory.

**Options**
`-b` queues the request to be executed in the background. The mstore command completes once the request has been queued or rejected. The default is to wait until the request fully completes or fails.

`-K keeptime`
specifies the length of time the file is to be kept online. If you specify the word `forever`, the file is always kept online. You may also specify a date, as `mm/dd/[yy]yy` or `dd-mon-[yy]yy`, or a time, as `hh:mm:ss` until which the file is to be kept online. Finally, you may specify a duration, as `nnnn[s m h d]`, for the number of seconds, minutes, hours, or days (the default) that the file must be kept online. The default is to keep the file online until it is not used for 24 hours. You may be restricted to certain `keeptime` values.

`-P purgetime`
specifies the length of time the file must be inactive before is can be removed from online space (i.e., purged). Specify a duration, as `nnnn[s m h d]`, for the number of seconds, minutes, hours, or days (the default) that the file must be unused before it is purged. The default is to not purge the file unless it has not been used for 24 hours. You may be restricted to certain `purgetime` values.

`-q` suppresses completion notification.

`-v` produces additional output during execution.
mstore get

**Success**

The command ends with a status code of 0.

**Failure**

The command ends with a non-zero status code.

**Notes**

1) Generally, the *path* should not be a glob. If you do specify a glob, the shell will expand the glob to a list of matching files. This list is probably not the one you wish to retrieve. Using globs may lead to unpredictable retrievals.

2) If *path* is a directory, an attempt is made to retrieve all of the files in that directory. The attempt fails if too many files need to be retrieved.

3) Each retrieved file has the same uid, gid, and permission bits that existed at the time that the file was copied into the Mass Storage System. However, **mstore** always turns off the execute and world-writable permission bits.

4) Recursive retrieval is not supported.

5) Each *path* must start with

   `/nfs/mstore/g/grpname` or `/nfs/mstore/u/username`

   and your *uid* must either be a member of a group matching *grpname* or must map to *username*. Otherwise, you must have read access to the file in order to retrieve the file.
5 mstore ls

```
mstore ls [ options ] [ path [ . . . ] ]
```

**options:** `-F`  `-l`  `-L limit`

**limit:**  `copy`  |  `failed`  |  `offline`  |  `online`  |  `pinned`

**Function**
List `mstore` managed files.

**Parameters**

`path` is the name of a directory or file to be listed. Optionally, specify one or more absolute `mstore` paths or paths relative to the current `mstore` directory. The default is to list the contents of the current directory, should it be an `mstore` managed directory.

**Options**

`-F` flags listed names with a special trailing character to indicate the entry’s properties. The following flags, from lowest to highest priority, are used:

/ - entry is a directory
< - file is online
$ - file is online and pinned
? - file is online but could not be migrated
: - file is saved twice on separate tapes\(^2\)

`-l` produces a long listing. The listing is virtually identical to that produced by the Unix `ls` command. See the notes for more information.

`-L limit` limits the listing to specific types of files:

- `copy` - only files designated as dual copy files
- `failed` - only files that could not be migrated
- `offline` - only files that are not online
- `online` - only files that are online
- `pinned` - only files that are pinned in the disk cache

By default, online and offline files are listed regardless of any other status (i.e., copy, failed or pinned).

---

\(^2\) This flag is temporal, the file may not have yet been copied offline at the time the `ls` command is issued.
mstore ls

Success
The command ends with a status code of 0.

Failure
The command ends with a non-zero status code.

Notes
1) Generally, the path should not be a glob. If you do specify a glob, the shell will expand the glob to a list of matching files. This list usually will not be representative of all stored files since some may not be online. Using globs may lead to unpredictable displays.
2) The output of mstore ls is essentially identical to that of the Unix ls command. Some differences exist:
   • The -F option adds flags for some additional properties.
   • The date field in the long listing varies, depending on the specified limit (-L option). When copy, online, offline, or the default is selected, the file’s modification date is displayed. When failed is selected, the time of the failure is displayed. When pinned is selected, the pin expiration time is displayed.
6 mstore put

\textbf{mstore put} \ [ options \ ] \textit{path} \ [ . . . ]

\textit{options:} \ -p \ -v

\textbf{Function}
Copies and, optionally, purges \texttt{mstore} managed files.

\textbf{Parameters}
\textit{path} is the name of the directory or file to be copied into the Mass Storage System and possibly purged. Specify at least one absolute \texttt{mstore} path or a path relative to the current \texttt{mstore} directory.

\textbf{Options}
\textbf{-p} \ \textit{purges} the file from online space after copying it into the Mass Storage System.
\textbf{-q} \ \textit{suppresses} completion notification.
\textbf{-v} \ \textit{produces} additional output during execution.

\textbf{Success}
The command ends with a status code of 0.

\textbf{Failure}
The command ends with a non-zero status code.

\textbf{Notes}
1) Generally, the \textit{path} should not be a glob. If you do specify a glob, the shell will expand the glob to a list of matching files; which may or may not be the files you really want written to the Mass Storage System.
2) Each path must start with
\texttt{/nfs/mstore/g/grpname or /nfs/mstore/u/username}
and your \texttt{uid} must either be a member of a group matching \texttt{grpname} or must map to \texttt{username}. 
7 mstore rename

```
  mstore rename [ options ] oldfn newfn

  options:  -f  -v
```

**Function**
Rename an mstore managed file.

**Parameters**

*oldfn*

is the name of the file to be renamed.

*newfn*

is the name *oldfn* is to have. The new name must not exist or must be a directory.

**Options**

* -f    quietly erases *newfn*, if it is a file, prior to renaming *oldfn* to *newfn*.

* -v    produces additional output during execution.

**Success**
The command ends with a status code of 0.

**Failure**
The command ends with a non-zero status code.

**Notes**

1) Neither *oldfn* nor *newfn* should be globs. If you do specify a glob, the shell will expand the glob to a list of matching files; which will likely cause the rename to fail.

2) Currently, you may not rename a directory. Only individual file renames are supported.

3) Each path must start with

   /nfs/mstore/g/grpname or /nfs/mstore/u/username

   and your *uid* must either be a member of a group matching *grpname* or must map to *username*.

4) You may not rename files across, in any combination, different user spaces and group spaces.
mstore rename

5) A file may lose its dual copy designation should it be effectively placed in a non-dual copy directory.
6) If newfn is a directory rename should proceed as if the user specified newfn-slash and appended the last component of oldfn.
7) If the target path does not exist, it is automatically created.
8) When a rename forces a file to be deleted, the deleted file must
   a. be owned by you or
   b. be assigned to a group in which you are a member and have the group write bit set (0020).
8 mstore set

```
mstore set [ options ] path [ . . . ]
```

**options:** `-g` `group` `-m` `mode`

**Function**
Set attributes for `mstore` managed files.

**Parameters**

`path` is the name the file whose attributes are to be set. Specify at least one absolute `mstore` path or a path relative to the current `mstore` directory.

**Options**

- `-g` `group`
  - is either an group id (i.e., gid) or a group name to be assigned to the file. You must be a member of the specified group and the file may not be in group space.

- `-m` `mode`
  - is either an absolute octal mode or a symbolic mode indicating the permission bits that are to be set for the file. See the notes for information about valid permissions.

**Success**
The command ends with a status code of 0.

**Failure**
The command ends with a non-zero status code.

**Notes**

1) Generally, the `path` should not be a glob. If you do specify a glob, the shell will expand the glob to a list of matching files. This list usually will not be representative of all stored files since some may not be online. Using globs may lead to unpredictable attribute modifications.

2) Each path must start with `nfs/mstore/g/grpname` or `nfs/mstore/u/username` and your `uid` must either be a member of a group matching `grpname` or must map to `username`. 
3) Absolute modes are composed of three octal digits. The first digit specifies the owner’s mode as 6xx (rw, the default), 4xx (r), or 2xx (w). The second digit specifies the group’s mode as x6x (rw), x4x (r, the default), x2x (w), or x0x (none). The third digit specifies the world’s mode as xx4 (r, the default) or xx0 (none). Extended permissions, execute permissions, and world write permissions are not respected. Additionally, the current umask value does not affect the resulting mode.

4) Symbolic modes follow the same rules as those defined for the Unix chmod command. Unlike absolute modes, the current umask value does affect the resulting mode. Otherwise, all other restrictions apply to the mode as described above.

5) Each modified file must
   a. be owned by you or
   b. be assigned to a group in which you are a member and have the group write bit set (0020).