Name

mtools - utilities to access DOS disks in Unix.

Introduction

Mtools is a collection of tools to allow Unix systems to manipulate MS-DOS files: read, write, and move around files on an MS-DOS file system (typically a floppy disk). Where reasonable, each program attempts to emulate the MS-DOS equivalent command. However, unnecessary restrictions and oddities of DOS are not emulated. For instance, it is possible to move subdirectories from one subdirectory to another.

Mtools is sufficient to give access to MS-DOS file systems. For instance, commands such as `mdir a:` work on the `a:` floppy without any preliminary mounting or initialization (assuming the default `'SYSCONFDIRmtools.conf'` works on your machine). With mtools, one can change floppies too without unmounting and mounting.

Where to get mtools

Mtools can be found at the following places (and their mirrors):

http://ftp.gnu.org/gnu/mtools/mtools-4.0.23.tar.gz
http://mtools.linux.lu/mtools-4.0.23.tar.gz
ftp://www.tux.org/pub/knaff/mtools/mtools-4.0.23.tar.gz
ftp://ibiblio.unc.edu/pub/Linux/utils/disk-management/mtools-4.0.23.tar.gz

Before reporting a bug, make sure that it has not yet been fixed in the Alpha patches which can be found at:

http://ftp.gnu.org/gnu/mtools/
http://mtools.linux.lu/
ftp://www.tux.org/pub/knaff/mtools

These patches are named `mtools-`*version–ddmm*`.taz`, where version stands for the base version, *dd* for the day and *mm* for the month. Due to a lack of space, I usually leave only the most recent patch.

There is an mtools mailing list at mtools @ tux.org . Please send all bug reports to this list. You may subscribe to the list by sending a message with 'subscribe mtools @ tux.org' in its body to majordomo @ tux.org . (N.B. Please remove the spaces around the "@" both times. I left them there in order to fool spambots.) Announcements of new mtools versions will also be sent to the list, in addition to the Linux announce newsgroups. The mailing list is archived at http://lists.gnu.org/pipermail/info-mtools/

Common features of all mtools commands

Options and filenames

MS-DOS filenames are composed of a drive letter followed by a colon, a subdirectory, and a filename. Only the filename part is mandatory, the drive letter and the subdirectory are optional. Filenames without a drive letter refer to Unix files. Subdirectory names can use either the '`/`' or '`\`' separator. The use of the '`\`' separator or wildcards requires the names to be enclosed in quotes to protect them from the shell. However, wildcards in Unix filenames should not be enclosed in quotes, because here we **want** the shell to expand them.

The regular expression "pattern matching" routines follow the Unix-style rules. For example, '`*`' matches all MS-DOS files in lieu of '`*.*`'. The archive, hidden, read-only and system attribute bits are ignored during pattern matching.

All options use the − (minus) as their first character, not `/` as you'd expect in MS-DOS.

Most mtools commands allow multiple filename parameters, which doesn't follow MS-DOS conventions, but which is more user-friendly.

Most mtools commands allow options that instruct them how to handle file name clashes. See section name clashes, for more details on these. All commands accept the `-V` flags which prints the version, and most accept the `-v` flag, which switches on verbose mode. In verbose mode, these commands print out the name of the MS-DOS files upon which they act, unless stated otherwise. See section Commands, for a description of the options which are specific to each command.

Drive letters

The meaning of the drive letters depends on the target architectures. However, on most target architectures, drive A is the first floppy drive, drive B is the second floppy drive (if available), drive J is a Jaz drive (if available), and drive Z is a Zip drive (if available). On those systems where the device name is derived from the SCSI id, the Jaz drive is assumed to be at SCSI target 4, and the Zip at SCSI target 5 (factory default settings). On Linux, both drives are assumed to be the second drive on the SCSI bus (/dev/sdb). The default settings can be changes using a configuration file (see section Configuration).

The drive letter : (colon) has a special meaning. It is used to access image files which are directly specified on the command line using the `-i` options.

Example:

```
mcopy -i my-image-file.bin ::file1 ::file2 .
```

This copies `file1` and `file2` from the image file (`my-image-file.bin`) to the `/tmp` directory.

You can also supply an offset within the image file by including `@@`*offset* into the file name.

Example:

```
mcopy -i my-image-file.bin@@1M ::file1 ::file2 .
```

This looks for the image at the offset of 1M in the file, rather than at its beginning.

Current working directory

The `mcd` command (‘mcd’) is used to establish the device and the current working directory (relative to the MS-DOS file system), otherwise the default is assumed to be `A:/`. However, unlike MS-DOS, there is only one working directory for all drives, and not one per drive.

VFAT-style long file names

This version of mtools supports VFAT style long filenames. If a Unix filename is too long to fit in a short DOS name, it is stored as a VFAT long name, and a companion short name is generated. This short name is what you see when you examine the disk with a pre-7.0 version of DOS.
The following table shows some examples of short names:

Long name	**MS-DOS name**	**Reason for the change**
---------	----------	--------------------
thisisatest	**THISIS~1**	**filename too long**
alain.knaff	**ALAIN~1.KNA**	**extension too long**
prn.txt	**PRN~1.TXT**	**PRN is a device name**
.abc	**ABC~1**	**null filename**
hot+cold	**HOT_CO~1**	**illegal character**

As you see, the following transformations happen to derive a short name:

- * Illegal characters are replaced by underscores. The illegal characters are `;+=[]',\"*\\<>/?:|`.
- * Extra dots, which cannot be interpreted as a main name/extension separator are removed
- * A ~*n* number is generated,

* The name is shortened so as to fit in the 8+3 limitation

The initial Unix-style file name (whether long or short) is also called the *primary* name, and the derived short name is also called the *secondary* name.

Example:

mcopy /etc/motd a:Reallylongname

Mtools creates a VFAT entry for Reallylongname, and uses REALLYLO as a short name. Reallylongname is the primary name, and REALLYLO is the secondary name.

mcopy /etc/motd a:motd

Motd fits into the DOS filename limits. Mtools doesn't need to derivate another name. Motd is the primary name, and there is no secondary name.

In a nutshell: The primary name is the long name, if one exists, or the short name if there is no long name.

Although VFAT is much more flexible than FAT, there are still names that are not acceptable, even in VFAT. There are still some illegal characters left (`\"*\\<>/?:|`), and device names are still reserved.

Unix name	**Long name**	**Reason for the change**
---------	**----------**	**--------------------**
prn	**prn-1**	**PRN is a device name**
ab:c	**ab_c-1**	**illegal character**

As you see, the following transformations happen if a long name is illegal:

* Illegal characters are replaces by underscores,

* A −*n* number is generated,

Name clashes

When writing a file to disk, its long name or short name may collide with an already existing file or directory. This may happen for all commands which create new directory entries, such as `mcopy`, `mmd`, `mren`, `mmove`. When a name clash happens, mtools asks you what it should do. It offers several choices:

`overwrite`
Overwrites the existing file. It is not possible to overwrite a directory with a file.

`rename`
Renames the newly created file. Mtools prompts for the new filename

`autorename`
Renames the newly created file. Mtools chooses a name by itself, without prompting

`skip` Gives up on this file, and moves on to the next (if any)

To chose one of these actions, type its first letter at the prompt. If you use a lower case letter, the action only applies for this file only, if you use an upper case letter, the action applies to all files, and you won't be prompted again.

You may also chose actions (for all files) on the command line, when invoking mtools:

`-D o` Overwrites primary names by default.

`-D O` Overwrites secondary names by default.

`-D r` Renames primary name by default.

`-D R` Renames secondary name by default.

`-D a` Autorenames primary name by default.

`-D A` Autorenames secondary name by default.

`-D s` Skip primary name by default.

`-D S` Skip secondary name by default.

`-D m` Ask user what to do with primary name.

`-D M` Ask user what to do with secondary name.

Note that for command line switches lower/upper differentiates between primary/secondary name whereas for interactive choices, lower/upper differentiates between just-this-time/always.

The primary name is the name as displayed in Windows 95 or Windows NT: i.e. the long name if it exists, and the short name otherwise. The secondary name is the "hidden" name, i.e. the short name if a long name exists.

By default, the user is prompted if the primary name clashes, and the secondary name is autorenamed.

If a name clash occurs in a Unix directory, mtools only asks whether to overwrite the file, or to skip it.

Case sensitivity of the VFAT file system

The VFAT file system is able to remember the case of the filenames. However, filenames which differ only in case are not allowed to coexist in the same directory. For example if you store a file called LongFileName on a VFAT file system, mdir shows this file as LongFileName, and not as Longfilename. However, if you then try to add LongFilename to the same directory, it is refused, because case is ignored for clash checks.

The VFAT file system allows you to store the case of a filename in the attribute byte, if all letters of the filename are the same case, and if all letters of the extension are the same case too. Mtools uses this information when displaying the files, and also to generate the Unix filename when mcopying to a Unix directory. This may have unexpected results when applied to files written using an pre-7.0 version of DOS: Indeed, the old style filenames map to all upper case. This is different from the behavior of the old version of mtools which used to generate lower case Unix filenames.

high capacity formats

Mtools supports a number of formats which allows storage of more data on disk as usual. Due to different operating system abilities, these formats are not supported on all operating systems. Mtools recognizes these formats transparently where supported.

In order to format these disks, you need to use an operating system specific tool. For Linux, suitable floppy tools can be found in the `fdutils` package at the following locations~:

```
ftp://www.tux.org/pub/knaff/fdutils/.
ftp://ibiblio.unc.edu/pub/Linux/utils/disk-management/fdutils-*
```

See the manual pages included in that package for further detail: Use `superformat` to format all formats except XDF, and use `xdfcopy` to format XDF.

More sectors

The oldest method of fitting more data on a disk is to use more sectors and more cylinders. Although the standard format uses 80 cylinders and 18 sectors (on a 3 1/2 high density disk), it is possible to use up to 83 cylinders (on most drives) and up to 21 sectors. This method allows to store up to 1743K on a 3 1/2 HD disk. However, 21 sector disks are twice as slow as the standard 18 sector disks because the sectors are packed so close together that we need to interleave them. This problem doesn't exist for 20 sector formats.

These formats are supported by numerous DOS shareware utilities such as `fdformat` and `vgacopy`. In his infinite hubris, Bill Gate$ believed that he invented this, and called it `'DMF disks'`, or `'Windows formatted disks'`. But in reality, it has already existed years before! Mtools supports these formats on Linux, on SunOS and on the DELL Unix PC.

Bigger sectors

By using bigger sectors it is possible to go beyond the capacity which can be obtained by the standard 512-byte sectors. This is because of the sector header. The sector header has the same size, regardless of how many data bytes are in the sector. Thus, we save some space by using *fewer*, but bigger sectors. For example, 1 sector of 4K only takes up header space once, whereas 8 sectors of 512 bytes have also 8 headers, for the same amount of useful data.

This method permits storage of up to 1992K on a 3 1/2 HD disk.

Mtools supports these formats only on Linux.

2m

The 2m format was originally invented by Ciriaco Garcia de Celis. It also uses bigger sectors than usual in order to fit more data on the disk. However, it uses the standard format (18 sectors of 512 bytes each) on the first cylinder, in order to make these disks easier to handle by DOS. Indeed this method allows you to have a standard sized boot sector, which contains a description of how the rest of the disk should be read.

However, the drawback of this is that the first cylinder can hold less data than the others. Unfortunately, DOS can only handle disks where each track contains the same amount of data. Thus 2m hides the fact that the first track contains less data by using a *shadow FAT*. (Usually, DOS stores the FAT in two identical copies, for additional safety. XDF stores only one copy, but tells DOS that it stores two. Thus the space that would be taken up by the second FAT copy is saved.) This also means that you should **never use a 2m disk to store anything else than a DOS file system**.

Mtools supports these formats only on Linux.

XDF

XDF is a high capacity format used by OS/2. It can hold 1840 K per disk. That's lower than the best 2m formats, but its main advantage is that it is fast: 600 milliseconds per track. That's faster than the 21 sector format, and almost as fast as the standard 18 sector format. In order to access these disks, make sure mtools has been compiled with XDF support, and set the `use_xdf` variable for the drive in the configuration file. See section Compiling mtools, and ‘miscellaneous variables’, for details on how to do this. Fast XDF access is only available for Linux kernels which are more recent than 1.1.34.

Mtools supports this format only on Linux.

Caution / Attention distributors: If mtools is compiled on a Linux kernel more recent than 1.3.34, it won't run on an older kernel. However, if it has been compiled on an older kernel, it still runs on a newer kernel, except that XDF access is slower. It is recommended that distribution authors only include mtools binaries compiled on kernels older than 1.3.34 until 2.0 comes out. When 2.0 will be out, mtools binaries compiled on newer kernels may (and should) be distributed. Mtools binaries compiled on kernels older than 1.3.34 won't run on any 2.1 kernel or later.

Exit codes

All the Mtools commands return 0 on success, 1 on utter failure, or 2 on partial failure. All the Mtools commands perform a few sanity checks before going ahead, to make sure that the disk is indeed an MS-DOS disk (as opposed to, say an ext2 or MINIX disk). These checks may reject partially corrupted disks, which might otherwise still be readable. To avoid these checks, set the MTOOLS_SKIP_CHECK environmental variable or the corresponding configuration file variable (see section global variables)

Bugs

An unfortunate side effect of not guessing the proper device (when multiple disk capacities are supported) is an occasional error message from the device driver. These can be safely ignored.

The fat checking code chokes on 1.72 Mb disks mformatted with pre-2.0.7 mtools. Set the environmental variable MTOOLS_FAT_COMPATIBILITY (or the corresponding configuration file variable, ‘global variables’) to bypass the fat checking.

See also

floppyd_installtest mattrib mbadblocks mcd mclasserase mcopy mdel mdeltree mdir mdu mformat minfo mkmanifest mlabel mmd mmount mmove mrd mren mshortname mshowfat mtoolstest mtype