MSH

COLLABORATORS

	TITLE : MSH	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MSH

1.1 MSH.guide

```
                MSH
***

This file documents MSH: 1.56.

Copyright (C) 1989-1996 by Olaf 'Rhialto' Seibert.  All Rights
Reserved. May not be used or copied without a license.

If you like the program, feel free to send me some $$, or even $$$ if
you feel like it. Please don't send any cheques not written in Dutch
guilders (standard abbreviation: nlg or hfl), since they cost a lot of
money (at least nlg 12.50; more than US$ 7) to cash.

All shareware resellers (and especially those in Germany) are reminded
that they are NOT allowed to make any profit off my work.  Resellers are
considered to make an illegal profit if they charge more for a disk than
Fred Fish does. Those resellers do NOT have a licence to use or copy
MSH. Also, if you distribute MSH:, you must distribute it in its
entirety: you are not allowed to leave parts out. This also includes the
source!



                Overview
                            What is MSH, and what is in the package?

                Requirements
                        What else do you need?

                Installation
                        How do you install it?

                Compatibility
                       How compatible is it?

                Utilities

                Suggestions
```

The messy file system handler is written by Sourcerer Olaf Rhialto Seibert.

1.2 MSH.guide/Overview

Overview
****

Mount MSH:

MSH:<any valid file specification>, or just

<any valid file specification>
if your current directory is somewhere on MSH:.

MSH: is a real Amiga-style File System handler that handles messydos formatted diskettes. You can use files on such messydos disks in almost exactly the same way as you use files on normal Amiga disks.

Supported are 40 or 80 tracks, double-sided 8, 9 or 10 sector double-density (DD) floppy disks, double-sided 15-21 sector high-density (HD) floppy disks, and harddisks with a 12 or 16-bit FAT of any dimension the FAT allows. I have not really tested MSH: on hard disks; proceed with extreme caution. I did test it in a large pseudo-partition though so I think that it should work.

See also the revision history, and
why pclones can't read Amiga disks.

1.3 MSH.guide/Requirements

Requirements
****

Besides what is included in this archive, you need AmigaDOS 1.2 or better. You may also need some knowledge of installing Mountlists and

.devices.

1.4 MSH.guide/Installation

```
                Installation
************
```

   Below is given a Mountlist entry.

   You can append this to your system's default Mountlist (DEVS:Mountlist), or to any other Mountlist you mention with the Mount command: Mount FROM SomeMountlist, as supported by AmigaDOS 1.3+ Mount commands.

   You must put the messydisk.device in DEVS:, and the file MessyFileSystem in L:. You can put these files in other locations, provided you adapt the file names in the Mountlist accordingly.

Mountlist
                What Mountlist to use

Installation Problems
   Problems? Naahh...

1.5 MSH.guide/Mountlist

```
                Mountlist
=========
```

   A sample Mountlist entry, that works with the AmigaDOS 3.0 Mount command:

   It will also work with other versions of the Mount command, but may need minor changes. For 1.2, change FileSystem to Handler. For 2.1+, leave out the 'MSH:' at the begin and '#' at the end, and put the remaining in a file called DEVS:DOSDrivers/MSH.)

```
/*
 *  Messy file system on messy blocks:
 */
MSH:    FileSystem = L:MessyFileSystem
        Device = messydisk.device
        Unit = 1
        Flags = 0
/*
 *  HighCyl is ignored, LowCyl, Surfaces, BlocksPerTrack
 *  and Reserved are used once to find the bootblock.
```

```
 *  Further parameters are found there.
 */
        LowCyl = 0 ; HighCyl = 79
        Reserved = 0
        Surfaces = 2
        BlocksPerTrack = 9
/*
 *  Initially the disk cache will be 5 sectors.
 */
        Buffers = 5
/*
 *  Our type of filesystem is 'MSD\0'.
 */
        DosType = 0x4D534400
/*
 *  If Interleave is set to 65536, MSH: will wait for DFx:
 *  (where x depends on the Unit parameter above) to recognize
 *  inserted disks (or not). See
                Compatibility
                .
 */
        Interleave = 65536
/*
 *  Use BufMemType = 3 [for MEMF_CHIP | MEMF_PUBLIC] if you
 *  use MSH: on the trackdisk.device.
 */
        BufMemType = 1
        BootPri = 0
        Stacksize = 3072
        Priority = 9
        GlobVec  = -1
/*
 *  Immediately load and start the MessyFileSystem
 *  when mounted.
 */
        Mount = 1
#
```

This Mountlist entry instructs MSH: to use floppy unit 1, i.e. DF1: and the messydisk.device. You may choose any name instead of MSH: that you like. For this case, MS1: would also be a sensible choice. You can read the separate manual page about messydisk.device. It is supposed to be possible to use MSH: on a harddisk by supplying the proper device name and unit number. If you wish to use floppies truly compatible with those messy machines, you should use the messydisk.device. You cannot change this name into something else. It is also supposed to work on 40-track drives, but this has not been tested by me.

The Mountlist entries LowCyl, Surfaces, BlocksPerTrack and Reserved are used once to determine the location of the messydos bootblock. Disk blocks of 512 bytes are assumed in this calculation. The block number of the bootblock is calculated as (LowCyl * Surfaces * BlocksPerTrack) + Reserved. Further information is then obtained from the bootblock (see below).

Additionally, these values, and HighCyl are used for default bootblocks, and for formatting.

MSH tries to be smart about HD floppies: if the specified number of sectors per track (spt) is greater than 10 (the maximum for DD disks), this spt is taken for HD floppies, and half the value for DD floppies. Otherwise, the spt is assumed to be for DD, and twice the value is set for HD floppies.

Every time you insert a disk, either the DD or HD spt value is made default (by messydisk.device) until the file system examines the boot block.

The DosType parameter indicates the type of file system that is implemented by this handler. As of AmigaOS 2.0, Commodore-Amiga defined a value for messy-type file systems: 0x4D534400 ('M' 'S' 'D' 0).

The Interleave parameter indicates whether MSH: should be nice to another file system which happens to use the same floppy drive. More about this below. Using floppies, you'll want to set Interleave to 65536, for use with harddisks 0 is needed. The lower 16 bits of Interleave are reserved to specify interleaving of sectors, but are currently ignored.

Older versions used DosType = 1, but this does not work anymore.

1.6 MSH.guide/Installation Problems

Installation Problems
==========

In the past, several problems have arised when installing MSH:. Most of them are:

* Not all Mount commands are the same. Not all keywords are always accepted, and some don't recognize keywords if they are preceeded by a TAB instead of spaces. Troublesome keywords are "Mount" and "FileSystem".

* Incorrect Mountlist entries can also be a problem. For example, it is easily forgotten to adjust the Unit value. This is especially difficult to detect if the Unit value refers to a drive that is not connected.

* Some Mount commands do funny things if the DosType parameter is missing. Not only does it default the value for DosType, but also for Stack, even if you do specify it. However this default of 3072 bytes is too small for some handlers, which causes mysterious dysfunctions.

1.7 MSH.guide/Compatibility

Compatibility
****

2.0+ Functions
Which 2.04+ functions are supported

Extensions
What useful extra things did I make?

Limitations
What does it not do?

General observations:

* Even though MSH: uses one of your floppy drives, you may still use that drive for normal Amiga disks. You will notice that when you insert a disk in the drive used by MSH:, both MSH: and the regular Amiga file system handler may attempt to identify the disk. The result will be that one of the two will be unable to read it, and then leave it alone so that the other may use it.

  The Interleave parameter indicates whether MSH: should be nice to another file system which happens to use the same floppy drive. Because this can only be DF0:, DF1:, DF2: or DF3:, MSH: can derive the name from the Unit parameter. If Interleave is set to 65536, MSH: will wait for DFx: (where x depends on the Unit parameter) to recognize inserted disks. If it recognizes an inserted disk as an AmigaDOS disk, MSH: will conclude that it should not even attempt to access the disk. If DFx: only notices that some disk has been inserted but does not think it is an AmigaDOS disk, MSH: goes ahead and tries to identify it. If DFx: has not reached some conclusion after approximately 7.5 seconds, MSH: does not wait any longer, and attempts identification anyway.

  If Interleave is not set, both MSH: and DFx: will try to access the disk at the same time. Only one of the two can succeed, but in the mean time they may have a violent fight about the control of the disk drive.

  For harddisk(partition)s, there is probably no other file system that MSH: must wait for, so Interleave should be set to 0.

* It is recommended that you don’t attempt to refer to MSH: while other filesystems are trying to read the disk.

* In some situations where there are multiple errors at the same time (such as trying to delete a non-existent file from a write-protected disk), the error reported may be different than the regular file system.

1.8 MSH.guide/2.0+ Functions

```
2.0+ Functions
==============

   The following functions from AmigaDOS 2.0 and higher are supported:

   SameLock(), SetFileSize(), ACTION_WRITE_PROTECT (the C:Lock command)
OpenFromLock(), IsFilesystem(), ChangeMode(), DupLockFromFH(),
ParentOfFH(), ExamineFH(), Format(), ACTION_SERIALIZE_DISK,
ACTION_GET_DISK_FFSM, ACTION_FREE_DISK_FFSM.

   Supported by emulation in dos.library:

   ExAll(), ExAllEnd().

   Not supported are:

   LockRecord(), UnLockRecord(), StartNotify(), EndNotify(),
MakeLink(), ReadLink().
```

1.9 MSH.guide/Extensions

```
                Extensions
==========



                 Character Conversions
                    Converting umlauts and double-s.

                 Extended Packets
                         Packets with extra functionality

                 Special Files
                            Files with a side effect (depecated)
```

1.10 MSH.guide/Character Conversions

```
                Character Conversions
---------------------

   MSH: has two character conversion tables installed, that can be used
while reading or writing a file. These tables convert the character
codes of the ü, ä, ö), Ü, Ä, Ö and ß characters between the
Amiga codes and PC or ST codes.

   To open a file with PC character conversion, add ";A" to the end of
the filename.  To open a file with ST character conversion, add ";B" to
```

the end of the filename.

Note: To specify a filename with a ";" in it on a CLI or Shell command line, you must enclose the entire name in quotes. This is because normally a semicolon starts a comment.

Note that no conversion between Amiga and PC or ST end-of-line characters is performed. Such a conversion would change the length of a file, and non-sequental access to a file would not be reliable. Furthermore, even the option of such a conversion would also slow down access to files which are not converted, which is the more common case.

A default conversion type, that applies to all files opened from then on, can be specified by accessing the "file" MSH::Cx, where x is the same character that specifies the conversion on file names.

To open a file without conversion, or to revert to default behaviour, use "@" as the conversion type.

See also

Special Files
,
Utilities
.

1.11 MSH.guide/Extended Packets

Extended Packets

The following packets have some extended functionality:

ACTION_DIE
The packet ACTION_DIE is supported, and even has the result of stopping MSH: and optionally unloading the file system code. This is convenient when you don't need to use MSH: for a while and helps you to conserve memory. A program called die is supplied that sends the ACTION_DIE packet to any handler you wish. (See

Utilities
) If there is any program running that expects MSH: to remain present even while there are no open or locked files on MSH:, it may be dangerous to remove MSH:. Unfortunately, one such program is the Workbench, so you should not kill MSH: if you also have the Workbench running.

ACTION_MORECACHE
MSH: handles the packet ACTION_MORECACHE (used by the AddBuffers command) and cache buffers more intelligently. When you add buffers, the required memory is not allocated immediately, but gradually as there becomes a use for it. When you remove a disk its buffers are no longer of use and therefore are all freed and returned to the free memory pool. And, what is more important, you

can decrease the maximum number of buffers. By specifying a negative number you reduce the number of buffers. If your AddBuffers does not like negative buffer counts, simply add 65536 to the (negative) number you actually intend. So, instead of -1 you can also specify 65535, -2 is 65534. You wouldn't want to add more than 32767 buffers at once anyway.

1.12 MSH.guide/Special Files

```
                Special Files
-------------

   DEPRECATED

   There are a few file names that have a special meaning. They all
start with MSH:: with two colons instead of MSH: with one colon.  For
some reason, some versions of AmigaDOS require you to use 3 colons
instead of 2.

   DEPRECATED

MSH::B
     By default, MSH: checks the bootblock to see if it contains a 8086
     or 68000 JMP instruction, or a 0 in the first byte. If so, it
     assumes that the bootblock is valid, and proceeds to fetch the
     information as detailed above. Then it checks if the values make
     any sense, in particular some minimum values are enforced. If this
     is not the case, all values are replaced with defaults. These
     defaults are based on a 720K disk with 2 FATS of 3 sectors (or
     whatever you specified in the Mountlist), clusters of 1K, and 112
     root directory entries.

     This default can be selected again by accessing the file
     "MSH::B7". (Note that the "B" must be in uppercase).

     An alternative behaviour is that disks with 'senseless' bootblocks
     are rejected instead of being 'fixed'. This can be selected by
     accessing the file "MSH::B3".

     Yet another option is that the bootblock is not checked at all,
     and the defaults are used in all cases.  This can be selected by
     accessing the file "MSH::B8".

     After accessing some MSH::B "file" a requester appears to confirm
     the value you selected. MSH: pauses until you click in the OK
     gadget.

     See also
                Boot block format
                .

MSH::F
     Some messydos computers have 80-track floppy drives, but the
```

FORMAT command does not know that. If you format an 80-track floppy on such a system, it is possible that only 40 tracks get formatted, and even the 'wrong' 40 tracks: the outer 40 tracks instead of every other track. If you insert such a disk in an MSH: drive, then MSH: knows that the disk uses only 40 tracks, but wrongly thinks it must use or simulate a 40 track drive. The result is that the files on that disk seem to contain garbage, or parts of other files. To correct this situation, you can access the special file name "MSH::F-" (the F must be in uppercase) each time after you insert that particular disk. On the other hand, the name "MSH::F+" forces MSH: to use 40 track drive emulation mode.

After accessing MSH::F+ or ::F- a requester appears to confirm the value you selected. MSH: pauses until you click in the OK gadget.

MSH::M
You can disable these requesters with MSH::M- and re-enable them with MSH::M+.

MSH::C
See
Character Conversions
,
Utilities
.

DEPRECATED

1.13 MSH.guide/Limitations

Limitations
==========

Due to the difference between messydos and AmigaDOS file systems, not all operations that are available for Amiga files can be applied to messydos files.

The following items are noteworthy:

File names
that silly 8+3 format

Special directories
"." and ".."

Filenotes
are impossible, of course

File protection bits
versus file attributes

```
Boot block format
           what is used of it

FAT

Workbench
                   cooperation with

Volume label

Volume and directory dates
  when are they updated and where
               are they stored

Multiple file systems
       how to have MS0: and MS1: etc.
```

1.14 MSH.guide/File names

```
File names
----------
```

The most eye-catching difference are the file names: messydos file names can be at most "8 + 3" characters long. This means that the "base" part of the file name may be eight characters long, maybe followed by a three-character "extension", which is separated from the base name with a period ('.'). Also, you cannot use all characters in the file names you wish, but the charater choice is basically limited to the uppercase letters A-Z, digits 0-9, and most "punctuation" characters except space * ? . < > / = | , : and \.

MSH: will try to map file names that you or a program attempts to use to names that are actually allowed. But for utmost flexibility, only the most basic limitations are enforced: the length of the name (8+3), uppercase alphabetics, and the presence of only one period in the name. This means that you might create file names that a messydos computer does not correctly know how to handle. But care has been taken that in no case it would be possible to create a file name that, once created, cannot be handled anymore by MSH:. You can always refer to that file with the same name you used before, even if MSH: changed it for you.

1.15 MSH.guide/Special directories

```
Special directories ('.' and '..')
-----------------------------------
```

On messydos floppies, every subdirectory has two special entries called "." and "..". They refer to the directory itself and its "parent"

directory, respectively. Programs that traverse directory trees on messydos systems take care to skip these "subdirectories", since they really are not "sub" direcories and recursively entering them would lead to nothing but trouble. (Their presence prevents the file system from having a tree structure.) Therefore, they are completely hidden from sight. You cannot see them in directory listings nor refer to them by any name. Nor will their dates/times be updated when they are changed in their respective primary directory entries.

But even while it is hidden, the ".." entry is of course still updated to reflect the new parent directory when a directory is renamed and moved. Note, by the way, that moving directories is an operation that is not supported by messydos itself.

Of course, the usual methods of accessing the current directory (the empty name "") and the parent directory ("/") do work.

It may be that in the future, when softlinks are supported better in AmigaDOS, these "." and ".." directories will be made visible again, but in the disguise of a soft link. In that case, programs which don't know how to handle them should either not exist anymore or ignore the links themselves.

1.16 MSH.guide/Filenotes

```
Filenotes (ACTION_SET_COMMENT)
------------------------------
```

Messydos has no provision for file notes and therefore they cannot be supported. However, to remain friends with the CLONE option of the C:Copy command, MSH: pretends success when setting a filenote.

1.17 MSH.guide/File protection bits

```
File protection bits vs. file attributes
----------------------------------------
```

Currently, the following interpretation of messydos file attributes is performed. The Directory and Volumelabel attributes are honored to make these directory entries different from plain files.

The HIDDEN and SYSTEM attributes are mapped to the H(idden) bit. The H bit is mapped to the HIDDEN attribute; the SYSTEM attribute cannot be changed.

The READ-ONLY attribute is mapped to the W(rite) and D(elete) bits. Inversely, if a file is set to either Write or Delete protected, the READ-ONLY attribute is set.

The ARCHIVE bit, when clear, sets the A(rchived) bit. On messydos the

ARCHIVE attribute means (when set) that the file MUST be backed up (by a backup program). A write to a file sets the ARCHIVE attribute. On AmigaDOS the Archived bit means (when set) that the file has been archived already. A write to a file clears the Archived protection bit.

1.18 MSH.guide/Boot block format

```
                Boot block format
-----------------
```

MSH: relies on the information in the boot block regarding the number of bytes per block, blocks per track, tracks per cylinder, blocks per cluster, etc. The physical information is not actually used to access the disk, except for the number of tracks which is derived from other information. It is used to decide if a 40 track disk is inserted in a 80 track drive. Single sided disks are not supported. The number of sectors may be whatever the underlying device driver supports. (With messydisk.device this is 8, 9 or 10; with trackdisk.device it is 11). Only the logical layout information is used: Size of a block, number of reserved blocks, number of FATs, number of blocks per FAT, number of root directory entries, number of blocks per cluster. The total number of blocks is used to determine the usable part of the disk. (If the FAT says "use a block beyond the limit", this error is not always detected).

If you format a disk on a messydos computer and wish to use it with MSH:, be sure to use a messydos version new enough to write this information in the boot block. (Version 3.2 works nicely for me.) If you wish to format such a disk on your Amiga, you can use the standard AmigaDOS Format command, or use the supplied messyfmt program.

See also

Special Files
.

1.19 MSH.guide/FAT

```
FAT
---
```

The FAT is currently not checked for integrity and consistency. The FAT is assumed to have 12-bit entries if the disk has at most 4086 ($FF6) clusters. If the disk has more clusters than that, 16 bit FAT entries are assumed. This is the method that seems to be used by messy dos 3.0.

1.20 MSH.guide/Workbench

```
Workbench
---------
```

The Workbench wants to use filenames that are impossible on messydos floppies: the icon files which end with .info.

To still make this possible, all filenames with the extension INF are converted to look like they end in .info. Which looks kind of silly, by the way, in directory listings.

1.21 MSH.guide/Volume label

```
                Volume label
------------
```

Normal AmigaDOS floppies have a part of the disk reserved for their name. With messydos floppies a name is optional, and should be in the directory in an entry with the special flag (attribute) Volumelabel. MSH: looks in the first directory block only to locate such a volume label. If found, it decides the name and creation date of the disk; if not found, the name will become "Unnamed" and the creation date will become 1-Jan-80. Note that AmigaDOS (and MSH:) keep disks apart by their name and creation date. As a result, all unlabeled disks, (and all disks with the same name) will look equivalent and cannot be distinguished. It is therefore strongly recommended that you provide every disk with a unique volume label; the standard AmigaDOS command Relabel can be used for this.

If you label a disk that had no label (in the first directory sector) before, MSH: will try to move a file from the first directory block to somewhere else in the root directory. If this is not possible, for instance because the root directory is full, then the labeling will fail as well.

The normal date that is contained in the volume node is the modification time of the root directory (so, if no label is present, the date is not kept either). The additional creation date is an nonstandard MSH feature, which may or may not be compatible with all versions of messydos. It will only be added when a new label is created; otherwise it is left undisturbed.

See also

Volume and directory dates

.

1.22 MSH.guide/Volume and directory dates

```
                Volume and directory dates
--------------------------
```

To recapulate the behaviour of directory and volume modification dates:

1. Directory modification dates are the dates in the normal directory entry that refers to that directory. This means that the date in "." and ".." entries is not updated. Since the root directory has no such directory entry, the volume label (if it exists) is used instead.

2. The volume creation date is put in some spare room in the volume label (the four unused bytes following the attribute byte).

3. If there is no volume label, the creation date is set to 1-1-1980, the first date allowed by messydos. The root directory modification date is only remembered as long as the disk is known to MSH; if it is later re-inserted it is reset to the initial date of 1-1-1980. The disk name is set to Unnamed. All such Unnamed disks look the same to AmigaDOS, and cannot be distinguished. NEVER have two such disks in use at the same time, or diaster will result.

See also

```
Volume label
.
```

1.23 MSH.guide/Multiple file systems

```
Multiple file systems
---------------------
```

The MessyFileSystem (versions 1.51 and higher) is "pure". Running two file systems from the same memory image is possible by using the separate utility MountShare. AmigaDOS 2.04 and before are not so smart to see that two file systems share the same FileSystem name in the Mountlist, so explicit use of MountShare is the only way to force sharing. See the appropriate documentation for details.

AmigaDOS 2.1 and higher has a Mount command that does this automatically by using the FileSystem.resource.

Therefore it is important the DosType value in the mountlist is set correctly.

1.24 MSH.guide/Utilities

```
Utilities
*********
```

* If you want to format default disks, and you are using system 2.0+, you can use the system standard Format command or the Workbench "Format Disk" menu. Default format means that you can only format disks with the same number of sectors as is specified in the Mountlist or mountfile (or twice that for HD floppies; see the MOUNTLIST section for details). A "quick" format is possible if the disk already was default-formatted and you only want to erase all files. A quick format on non-default formatted disks will appear normal but may result in loss of data at a later time.

  If you want to format a disk in a special way you can always use one of the special messy formatting utilities as mentioned below.

  * To format your disks you can use MSH-Format (if you are running AmigaOS 2.04 or greater). This is a very intuitive program so you just need to point and click to format disks. You can set all kinds of parameters, if you need to, but not all combinations of them result in usable disks.

  * There is a commodity called MSH-Set (if you are running AmigaOS 2.04 or greater). This is a very intuitive program to set options in MSH. It supports the usual commodity ToolTypes CX_PRIORITY (default 0), CX_POPKEY (default rawkey ctrl alt m) and CX_POPUP (default YES). Should accidentally the commodities exchange libary not be available, then this commodity still runs as an ordinary program without commodity features.

    For an explanation of the flags, it is best to refer to the (hereby obsolete) MSH:: commands. Furthermore, the function of LoadConv (see below) is contained in MSH-Set.

  * A program called MessyFmt allows you to format disks in messydos format. It runs on all versions of AmigaDOS.

    Usage: messyfmt [ASK/BOOT/QUICK] <unitnr> [<device>]

    Formats a messydos volume in any desired shape.

    The items between [brackets] are optional.

    You can specify one of ASK, BOOT or QUICK.

    The <device> name defaults to "messydisk.device".

    Example: The command "messyfmt 1" will format the disk in floppy disk unit #1, also used for DF1:.

    If you specify QUICK, the tracks with the bootblock, the FAT and the root directory are formatted, but the rest of the disk is untouched.

If you specify BOOT, only the bootblock is written. The disk must already be formatted in messydos format for this to work.

If you do not specify any of these options, the whole disk is formatted with all default parameters.

If you specify ASK, the program asks for all parameters to put in the bootblock, like

Bytes per sector? [512]

If you just hit return you accept the default value, otherwise you can enter a new value in the usual notation: 0x = hexadecimal (the digits A-F must be in upper case), 0 = octal, and everything else is decimal. A few suffixes are allowed as multiplication factor: m = mega (1048576), k = kilo (1024), s = sectors (512) and b = byte (1). (Not all terms make sense with all questions.) You may create 8, 9 or 10 sector disks by answering the question "Sectors per track" appropriately. This value is stored in the boot block, and also used in the format command for the disk device.

You are also asked if you wish to format the whole disk, and if not, if you want to write only the bootblock, or the bootblock, FAT and directory.

Finally, to see if you are still awake, you must enter 42 if you are sure to do it.

After formatting has completed, you must remove the disk from the drive, in order to let all involved filesystems know that the disk just died and reincarnated.

Hint: If you want to format a 40 track floppy in an 80 track drive, it is probably easiest to first format it with 80 tracks, and then re-format just the bootblock. Only then you need to give the correct values for 40 tracks.

* Die allows you (under certain circumstances) to terminate the MSH: process and unload the MessyFileSystem from memory.

Usage: die [unshare|unload] MSH-DEV:

So if you have mounted MSH as MSH:, and use "Die MSH:", an ACTION_DIE packet is sent. MSH then checks if any files or locks are still outstanding. If there are none, termination is possible.

The optional argument unload has the effect that the DIE packet is tagged specifically so that MSH knows that it should unload itself. Sometimes this is not a safe operation, therefore the extra reminder is needed.

If you used MountShare (see appropriate documentation), you should specify unshare for all devices but the last one, and then unload for the last one, if you have checked that it is really the last one. You can check this with the C:Info command: if only the last

device is shown it is safe to specify unload. The effect of the unshare option is that MSH: will erase the SegList pointer from its DeviceNode, which was put there by MountShare.

* Not only can character conversions be performed on files read and written, the conversion table can even be user-defined with help of the program LoadConv.

  Usage: loadconv MSH-DEV: X file

  where X is the conversion id (A or B).

  From then on, all reads on files opened as "file;X" use the new conversion.

  The file must contain exactly 512 bytes of data. The first 256 bytes form the conversion table used when writing files, the second 256 bytes are the reading conversion table. This table simply contains the converted values of the characters 0 upto 255, i.e. if you want Amiga character number 13 to be translated into other character 10, the write table contains a 10 in position 13, and the read table contains a 13 in position 10.

* In order to help making your own conversion tables, a rexx script called MakeConv.rexx is supplied.

  Usage: [rx] makeconv filename [amiga-char other-char]*

  The first argument is the filename in which the table is to be written. Next are any number of pairs (including none) of characters, the first is the Amiga character, the second the converted one. The conversion tables are then constructed such that they are each other's inverse, as long as this is possible. For instance, this is impossible if you issue the command

  rx makeconv test 1 2 1 3

  The tables are updated left-to-right, so that the result will be that Amiga-1 is converted into Other-3, and Other-2 and -3 both into Amiga-1.

  You can specify the characters in decimal (5), hexadecimal ($78 or 0xAB), or as characters ('a or "z").

  There are 2 additional rexx programs, which use makeconv, and allow you to create the equivalent of the default "A" and "B" conversion tables. They are called MakePC.rexx and MakeST.rexx, and create files called PC-conv.msh and ST-conv.msh. You can adapt these to generate more elaborate conversions. These commands take no arguments.

* DiskCopy can be used to copy messydos disks, provided you have one or more file systems mounted.

1.25 MSH.guide/Suggestions

```
Suggestions
***********

   The things I mention here are merely interesting suggestions. They
have not all been tried. If they work for you, or if you think of some
more, please let me know.

   Since you can use any disk-type device under MSH:, you need not
necessarily use messydisk.device, but you could use trackdisk.device
(the normal Amiga disk driver), some kind of harddisk.device or
jdisk.device (for use with messydos partitions on bridgeboards), or
Matt Dillons fmsdisk.device (a file disguised as a disk device, if you
have a messydos disk in an AmigaDOS file for JLink). I have heard of a
case where an Amiga drive with MSH: was accessed from the BridgeBoard
through JLink.
```

1.26 MSH.guide/Bugs

```
                Bugs
****

   With AmigaOS 2.00 - 2.03, versions prior to 1.40 are unable to write
to disks because messydisk.device can't do this for some reason. 2.04
does not pose any problems for those very old versions, as far as I
know.

   See also
               Compatibility
               ,
               2.0+ Functions
               ,
               Extensions
               ,
               Limitations
               .
```

1.27 MSH.guide/Glossary

```
Glossary
********

   This chapter deliberately left empty.

FAT
     File Allocation Table. Messydos' silly way of recording where the
     clusters for each file are.
```

cluster
     The minimum allocation unit for files and subdirectories. A
     cluster may be a power of two, and is expressed in sectors.

volume label
     The silly way of storing the name of the disk. Not all disks have
     one.

1.28 MSH.guide/Index

Index
*****