Fdutils

Linux floppy utilities
Edition 5.5, for Fdutils version 5.5 (CPM4L 1.0 5.5-20050323)
Mar 2005

by Alain Knaff

General Introduction

Fdutils is a collection of utililties for configuring and using the Linux floppy driver. With fdutils, you can:

1. Format disks with a higher capacity than usual (up to 1992KB on a 3 1/2 HD disk).
2. Reconfigure the autodetection sequence to automatically detect some of these extended formats.
3. Access various internal driver structures and drive configuration using the floppycontrol program.

This manual describes these utilities, and also the floppy driver itself.

1 Where to get fdutils and its documentation

Fdutils can be found at the following places (+ mirrors):

```
ftp://www.tux.org/pub/knaff/fdutils/fdutils-5.5.tar.gz
ftp://metalab.unc.edu/pub/Linux/utils/disk-management/fdutils-5.5.tar.gz
ftp://tsx-11.mit.edu/pub/linux/sources/sbin/fdutils-5.5.tar.gz
```

The FAQ included in this package is also available separetely at:

```
http://alain.knaff.linux.lu/floppy/FAQ.html
http://www.tux.org/pub/knaff/floppy/FAQ.html
```

The FAQ at fdutils.linux.lu and www.tux.org is usually more up to date than versions found elsewhere. Thus, if you don't find an answer in the copy of the FAQ you have, please check this one for more recent info.

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

```
http://fdutils.linux.lu
ftp://www.tux.org/pub/knaff/fdutils
```

These patches are named `fdutils-`*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 fdutils mailing list at fdutils `@` www.tux.org . Please send all bug reports to this list. You may subscribe to the list by sending a message with 'subscribe fdutils `@` www.tux.org' in its body to majordomo `@` www.tux.org . (N.B. Please remove the spaces around the `"@"` both times. I left them there in order to fool spambots.) Announcements of new fdutils versions will also be sent to the list, in addition to the linux announce newsgroups. The mailing list is archived at http://www.tux.org/hypermail/fdutils/latest

2 Basic usage

This chapter describes basic usage of floppies, and gives a few simple tips for using floppies under Linux.

2.1 How disks are organized

All floppies have two levels of *formatting*, both of which must be known in order to read them. The first is the *binary* or *sector* level format, which is how raw data is stored on the disk. The second is a higher level organization, often called a *file system*, which allows multiple files to be conveniently stored on the disk.

For example, a typical 1.44MB disk contains a low-level format, with 18 sectors per track, 80 tracks, and two sides (or heads); each sector can hold 512 bytes of data for a total of 1474560 bytes (or 1440 KB). When used under MS-DOS, this floppy would have a small portion of the disk used to keep track of files on the disk (including a bootsector, file allocation tables, directories, etc.).

The floppy driver generally takes care of reading the binary, or low-level format. It can often "guess" the low-level disk geometry needed to read the disk. This is called autodetection (see Chapter 7 [Autodetection], page 20). If the driver can't autodetect the disk (e.g. if it is in an unusual format) you can tell the driver what the geometry is either by using the `setfdprm` (see Section 10.9 [setfdprm], page 47) utility or by using a fixed geometry device device (e.g. '`/dev/fd0H1440`').

Under Linux, many different file systems from many sources can be used. Some of these file systems are interpreted via a utility program (for example `mtools` for using disks with an MS-DOS file system). Many file systems can alternatively be "mounted" to appear in the UNIX directory structure until subsequently being unmounted; this is usually implemented by having the kernel itself interpret the file system on the disk.

2.2 File systems supported by Linux

The following file systems are supported:

OS/2 HPFS:
: read-only support (mount/kernel)

Mac HPFS 1.44MB:
: read-only (xhfs utility)

MS-DOS: read, write, format (mtools utility *and* mount/kernel)

tar, cpio: compatible with many variations of UNIX (tar, cpio utilities)

System V, minix, xia, ext, ext2:
: (mount/kernel)

pure binary disk access:
: no file system (any program, usually dd, cat, and cp)

2.3 What's in a name

The following figure shows the meaning of the different parts of the name of a floppy device:

```
 +--------------- /dev: directory for devices
 |   +------------- fd: floppy disk device prefix
 |   | +------------ 0: floppy drive #0 (A:) (0-1 typical, 0-7
 |   | |               possible)
 |   | |+-- 3.5" drive: (use d for 5.25" double density drives, and
 |   | ||                    h for 5.25" high density drives,
 |   | ||                    u for 3.5" drive of any density)
 |   | ||   +---- 1440: Capacity (in KB) of format (usually between
 |   | ||   |           360 and 3920)
/dev/fd0u1440
```

2.4 What to do if you get an unidentified floppy disk

```
dd if=/dev/fd0 of=/tmp/foo count=1
        # If it works:
getfdprm                # This will report what geometry the disk has
file /tmp/foo           # This may indicate the type of file system
mdir a:                 # Check for an MS-DOS file system
tar tvf /dev/fd0        # Check for a tar archive
cpio -itv < /dev/fd0    # Check for a cpio archive
e2fsck /dev/fd0         # Check for an "ext2" file system
        # If it doesn't work:
        # Try the above dd command using various /dev/fd0* devices
```

2.5 Nickel tours

2.5.1 mtools

```
mdir a:                # Read directory of MS-DOS disk in drive A:
mcopy /tmp/foo\* a:    # Copy files beginning with foo in /tmp to A:
mcopy a:\* .           # Copy all files from A: to current directory
mformat a:             # Add MS-DOS file system to formatted disk
```

2.5.2 Tar (Tape ARchive)

```
tar tvf /dev/fd0              # Read directory of tar archive in
                              # drive A:
tar cvf /dev/fd0 foo1 foo2    # Write foo1 and foo2 to A: in tar
                              # format foo1/foo2 can be entire
                              # directory trees
tar xvfp /dev/fd0             # extract entire tar archive in
```

```
                                # drive A:
```

Tar is not a file system. Only low-level format (`superformat`, see Section 10.10 [superformat], page 48) are needed to prepare a disk to accept a tar archive.

2.5.3 CPIO (CoPy In/Out)

```
cpio -itv < /dev/fd0        # Read directory of cpio archive in A:
find foo1 foo2 -print | cpio -ov < /dev/fd0
                            # Write foo1/foo2 to A:
                            # foo1/foo2 can be entire directory trees
cpio -idumv < /dev/fd0      # extract entire CPIO archive in drive A:
```

Note: blocks reported are in 512-byte units (due to UNIX System V heritage). Cpio is not a file system. Only low-level format (fdformat or superformat (see Section 10.10 [superformat], page 48) needed.

2.6 Ext2 (Second Extended File System)

```
mke2fs /dev/fd0 1440          # Makes an ext2 filesystem of 1440
                              # block on A:
mke2fs -c /dev/fd0 1440       # Same as above, but tests floppy first
e2fsck /dev/fd0               # Tests filesystem integrity. (like
                              # chkdsk in Dos)
e2fsck -p /dev/fd0            # Repairs filesystem. (like chkdsk /f
                              # in Dos)
mount -t ext2 /dev/fd0 /mnt   # Mounts the disk in A: on /mnt.
                              # The directory /mnt must already exist
umount /mnt                   # Unmounts /mnt. No process should
                              # have its working directory in /mnt
                              # No process should have open files in
                              # /mnt
```

Note: don't use ext2 on 2m disks On some systems `mke2fs` is also called mkfs.ext2, and e2fsck is also called fsck.ext2

2.7 New Features of 1.2+ kernels

2.7.1 New features of 1.2+ kernels

- Faster and more comprehensive automatic sensing of floppy formats
- Second Floppy Disk Controller (FDC) supported
- DOS fdformat-style formats (up to 21 sectors on HD 3.5" disk)
- DOS 2m-style formats (up to 24 sectors equivalent on HD 3.5" disk)
- non-DOS 2m-inspired formats
- Several long-standing bugs fixed

- More exact detection of FDC type
- More exact detection of floppy drives

2.7.2 New features of mtools-3.0

- Support for new floppy formats (fdformat, 2m, 2m-like, ED)
- 2.88MB (Extra Density) floppies supported
- More friendly syntax (e.g. "mcopy a:", "mmove")
- Improved mmount
- 16-bit FATs (needed for some ED formats)
- Automatically sets disk geometry for Linux
- Several bug fixes

NOTE: Mtools has no longer maintained by its original maintainer Emmet P. Gray after 2.0.7.

2.7.3 New Utilities

- `superformat` (replaces fdformat; up to 3.84 MB floppies, faster, calls mformat)
- new `getfdprm/setfdprm`
- `fdrawcmd` (allows user-mode programs to do low-level floppy actions) `floppycontrol` (general-purpose floppy driver configuration utility)
- `MAKEFLOPPIES` (makes floppy devices)

3 Device numbers

The floppy device nodes are usually made using the `MAKEFLOPPIES` shell script (See Section 10.8 [makefloppies], page 46.).

The major device number for the floppy drives is 2. The minor device number contains describes which drive it represents, and may in addition describe the kind of media which is currently in the drive.

There are two kind of floppy devices:

- Variable geometry device nodes. Their minor number doesn't depend on the media in the drive, and is calculated as follows:

  minor_device = 128 * *fdc_nr* + *unit_nr*

- Fixed geometry device nodes. Their minor number not only depends on the drive which they represent, but also the type of media currently in the drive. It is computed as follows:

  minor_device = 128 * *fdc_nr* + *unit_nr* + 4 * *format_nr*

In this formula, *fdc_nr* is the number of the floppy disk controller (0 or 1, usually 0), and *unit_nr* is the Unit number (0 to 3, 0 for Dos drive `A:`, and 1 for Dos drive `B:`). *Format_nr* is only meaningful for the fixed format devices. It describes the disk geometry that is used. It is an index into the *geometry list* Section 3.3 [geometry list], page 6. Using all available controller numbers and all available drive numbers, you may thus connect up to 8 floppy drives to a single Linux box.

3.1 Variable format devices

Variable format devices don't have an intrinsic geometry. When using these devices, the geometry has to be set either by using autodetection (see Chapter 7 [Autodetection], page 20), or by using the `FDSETPRM` or `FDGETPRM` ioctl. The latter ioctl can be issued using the `setfdprm` (see Section 10.9 [setfdprm], page 47) and `getfdrpm` (see Section 10.7 [getfdprm], page 46) programs. With the default settings, common formats are detected transparently, and you can access any disk transparently using the variable format devices.

The geometry information is kept as long as the disk is in the drive, and is discarded as soon as the disk is removed, unless the geometry has been declared *permanent* by using setfdprm's `-p` flag (see Section 10.9 [setfdprm], page 47).

3.2 Fixed format devices

Fixed format devices should not be used under normal circumstances.

Fixed format devices have an intrinsic geometry. They are useful for the `fdformat` program (which is now considered obsolete), and for booting off floppies which have formats that are different from the default format (because during booting, there is no application that can issue the otherwise needed `FDSETPRM` ioctl).

3.3 The geometry list

The floppy driver contains a builtin list of 32 formats. This list is used for two purposes:

- It says which geometry is used for the *fixed format* devices.
- It is used for *autodetection*

The following formats (geometries) are known:

format_nr	**Format**
0	autodetect
1	360KB, 5.25" DD drive
2	1200KB, 5.25" HD drive
3	360KB, 3.5" DD drive
4	720KB, 3.5" DD drive
5	360KB, 5.25" DD disk in HD drive
6	720KB, 5.25" DD disk in HD drive
7	1440KB, 3.5" HD drive
8	2880KB, 3.5" ED drive
9	3120KB, 3.5" ED drive
10	1440KB, 5.25" HD drive
11	1680KB, 3.5" HD drive

12	410KB, 5.25" DD disk in HD drive
13	820KB, 3.5" DD drive
14	1476KB, 5.25" HD drive
15	1722KB, 3.5" HD drive
16	420KB, 5.25" DD disk in HD drive
17	830KB, 3.5" DD drive
18	1494KB, 5.25" HD drive
19	1743KB, 3.5" HD drive
20	880KB, 5.25" DD drive
21	1040KB, 3.5" DD drive
22	1120KB, 3.5" DD drive
23	1600KB, 5.25" HD drive
24	1760KB, 3.5" HD drive
25	1920KB, 3.5" HD drive
26	3200KB, 3.5" ED drive
27	3520KB, 3.5" ED drive
28	3840KB, 3.5" ED drive
29	1840KB, 3.5" HD drive
30	800KB, 3.5" DD drive
31	1600KB, 3.5" HD drive

This table lists first the format_nr (0-31) used to compute the minor number, then the capacity of the format (360KB - 3200KB), and then the type of the drive in which this format is used.

The formats 0..8 are the standard PC formats. The remaining formats are extended capacity formats. Some of them have been taken from Heiko Schroeder's fdpatches (after correcting some minor bugs). Others have been added by David Niemi and me (Alain Knaff). Formats 9, 12, 13, 16, 17, 30 and 31 are non-interleaved formats with normal sized sectors, and have the highest capacity that can be achieved without resorting to interleaving or bigger sectors (Section 6.1 [More sectors], page 14). Formats 10, 11, 14, 15, 18, 19 use interleaving interleaving to achieve a higher capacity (Section 6.2 [Interleave], page 14). Formats 20 and 22 to 29 use bigger sectors than usual (Section 6.5 [Larger sectors], page 16 and Section 6.6 [Mixed size sectors], page 17).

In addition to these techniques, formats 13-19 use more cylinders than usual (Section 6.4 [More cylinders], page 16). **USE THESE FORMATS (13-19) ONLY IF YOUR DRIVE SUPPORTS THE NECESSARY NUMBER OF TRACKS**

3.4 Adding new formats

You can redefine the default formats using the setfdprm program (Section 10.9 [setfdprm], page 47)[1]. The following example illustrates how to add a new 19 sector format, and make a device entry for it. First, we pick an entry for it, which we want to reuse. I recommend to redefine an entry which is only rarely used. For instance, if you have no 5 1/4 drive on your system, you can redefine any 5 1/4 entry without a loss. In our example, we pick 10.

First we make the device node:

```
mknod /dev/fd0H1520 b 2 40
          ^         ^ ^  ^
          |         | |  Minor device number (format number * 4 +
          |         | |                       drive + controller*128)
          |         | Major device number (always 2!)
          |         Blockdevice
      A name that you choose for the format. I
      recommend to base the name on the capacity,
      but you may choose any name you want.
```

Then we redefine the geometry of the new device:

```
setfdprm /dev/fd0H1520 1520 19 2 80 0 0x1b 0 0xcf 0x6c
```

Note: This redefines the geometry for any device node with the same format number, not just the new node.

The new geometry is only valid until the next reboot (or removal of the floppy module). In order to make it permanent, you have to execute the `setfdprm` command from your '`/etc/rc`' file or whenever you insert the floppy module.

4 Media description

4.1 Introduction

Fdutils-5.0 introduces a new uniform format description, which is supported both by setfdprm and superformat. The new format description is easyer to handle, because it allows to set the different parameters of a format description in a symbolic and position independant way, using a series of *variable=value* clauses. Moreover, it automatically fills in sensible default values for unspecified parameters. Thus you only need to describe those aspects of the format that are important to you, and let the system handle the others.

Moreover, the new description separates those aspects that were specific to the drive (like for instance its rotation speed) from those that are specific to the media (spacial density, number of sectors, etc.).

The same description can be used both by setfdprm and superformat:

[1] In that case, the `MAKEFLOPPIES` program (Section 10.8 [makefloppies], page 46) no longer works to generate a correct name for these formats, and you have to make them manually.

```
setfdprm /dev/fd0 hd sect=21 cyl=83
superformat /dev/fd0 hd sect=21 cyl=83
```

The first line above configures a 21 sector/83 cylinder format for drive 0, and the second line formats a disk using this same format.

4.2 Syntax

A media description is a series of *variable=value* and *selector* clauses. *Value* is a number followed by an optional unit. The unit is either `KB` (1024 bytes) or `b` (blocks of 512 bytes), or none (bytes).

4.2.1 Selecting the density

To select a density just insert its two letter code into the format description. Selecting a density also selects its default number of sectors, heads and cylinders. However, these latter parameters can be overridden.

`hd` High density (1440KB for 3 1/2 and 1200KB for 5 1/4). The most commonly used format today.

`dd` Double density (720KB for 3 1/2 and 360KB for 5 1/4)

`ed` Extra density (2880KB for 3 1/2)

`qd` Quad density (720KB for 5 1/4).

`sd` Single density (no nominal size). Used mostly for CP/M. Only for experienced users.

If no density is given, the maximal density supported by the drive is used. However, in order to keep the drive description and the media description independent, I strongly suggest that you **always** indicate the density anyways.

4.2.2 Selecting the number of cylinders, heads and sectors

This subsection describes how to select custom formats with a non-standard number of heads, cylinders or sectors. However, note that just describing the number of sectors, heads and cylinders is not enough: you also need to indicate which density your custom format is based on (cf. previous section).

`sect=`*`nb_of_sectors`*
: This describes the number of sectors.

`head=`*`nb_of_heads`*
: This describes the number of heads to be used.

`cyl=`*`nb_of_cylinders`*
: This described the number of cylinders to be used.

4.2.3 Selecting non-standard sector sizes

In order to achieve a higher capacity, you may want to use a bigger sector size.

`ssize=`*`sector_size`*
: Choses a bigger sector size. The sector size is expressed in bytes. Only powers of two between 128 and 32768 are acceptable

`sect=`*`nb_of_sectors`*
: Describes the number of sectors. For example `hd sect=11 ssize=1024` describes a format where one track (1 side) is made up of 11 sectors of 1024 bytes each (thus 11KB per track, and 22KB per cylinder).

`tracksize=`*`size_of_one_track`*
: Describes the size of one track. For example, `hd tracksize=11KB ssize=1KB` describes a format where one track contains 11KB of data (tracksize) stored in sectors of 1KB each.

  This option exists mainly to describe MSS (mixed sector size) formats. For example, `hd tracksize=12KB mss` describes a format where one track which contains 12 KB of data. The sectors size are chosen by the system in a way to take up the least raw space: 8KB + 4KB.

`mss`
: This option says that the format is an MSS format.

`2m`
: This option says that the format is a so-called 2M format. These formats are intended for easy readability on DOS boxes. Their first track has the usual 18 sectors, whereas the other tracks have bigger sector, and in some cases mixed sector sizes.

4.2.4 Legacy formats

The `swapsides` format allows to descibe disks whose sides are swapped, such as CBM1581 disks.

4.2.5 Expert options

The following options are not needed in most common situations, as they are implied by the *density* selector. They may be needed to read some legacy (CP/M) formats.

`tpi=48` For 5 1/4 disks only. This says that the format uses double-spaced cylinders (implied by double density).

`tpi=96` For 5 1/4 disks only. This says that the format uses single-spaced cylinders (implied by quad and high density).

`fm=0` Uses MFM encoding (implied by double, quad, high and extra density)

`fm=1` Uses FM encoding (implied by single density)

`dtr=`*`dtr-code`*

Sets the data transfer rate. The following table lists the dtr codes for various transfer rates:

dtr-code	rate for FM	rate for MFM
0	250kb/s	500kb/s
1	150kb/s	300kb/s
2	125kb/s	250kb/s
3	500kb/s	1000kb/s

`perp=0` Do not use "perpendicular mode" sector headers (this setting is implied by single, double, quad and high density).

`perp=1` Use "perpendicular" sector headers (this setting is implied by extra-density)

`gap=`*`value`*

Sets the size of the read/write gap. I don't know the purpose of this parameter (which is passed *as-is* to the floppy controller): any value seems to work with any format...

`fmt_gap=`*`value`*

Sets the size of the formatting gap. This is only used by the now obsolete `fdformat` program, and not by superformat.

4.3 The media description dictionary in /etc/mediaprm

'`/etc/mediaprm`'[2] contains a dictionary of commonly used media descriptions. Each description is identified by a name, which can then be used by setfdprm or superformat to refer to it, instead of an explicit description.

Each definition starts with "*name*":, followed by the actual description. Definitions may be spread over several lines, for better readability. The file may contain comments, which start with # and stop at the end of the line.

5 Drive descriptions

Unlike earlyer version, fdutils-5.0 separates drive descriptions and media description. For more details on this separation, see Section 4.1 [Introduction (Mediaprm)], page 8. Drive descriptions are used to describe the hardware characteristics of a drive, such as their maximal density, their rotation speed, their form factor, etc.

5.1 Syntax

A drive description is a series of *variable=value* and *selector* clauses.

[2] The actual location of this file depends on the value of the `sysconfdir` compile time configuration variable (see Chapter 11 [Compile-time configuration], page 53 for details)

5.1.1 Density

The density of a drive is the highest media density that it supports. Density is one of `sd`, `dd`, `qd`, `hd` or `ed`. Usually, you do not need to specify this parameter, as it can be derived from the drives CMOS code.

5.1.2 Form factor

The form factor of a drive describes the physical dimensions of the media it accepts. It is one of `3.5`, `5.25` or `8`. Usually, you do not need to specify this parameter, as it can be derived from the drives CMOS code.

5.1.3 Cmos code

The PC Bios already knows on its own about the most common drive types. These are named by an integer from 1 to 6, according to the following table.

```
0       no drive installed
1       5.25 DD
2       5.25 HD
3       3.5 DD
4       3.5 HD
5       3.5 ED
6       3.5 ED
```

As you see 3.5 ED drives have two possible codes. Some BIOSes use 5, others use 6. The reason for this is that initially 5 was intended for floppy tape drives, and only 6 was for 3.5 ED drives. However, some BIOS manufacturers didn't know about this convention, and used 5 for the then "new" 3.5 ED drives.

Usually, you do not need to specify this parameter, as it can be read from the physical CMOS of your PC. This parameter may be useful if your BIOS does not store the drive's CMOS code at the expected place, or if you have more than two drives.

5.1.4 Other parameters

`deviation=`*`deviation`*

Tells how much more/less raw capacity the drive has than the standard. Due to slightly different rotation speeds[3] and to slightly different data transfer rates, the raw capacity per track can vary slightly. For normal formats, these small deviations from the prescribed raw capacity is not harmful, as these have plenty of safety margins built in. However, the new extra capacity formats are affected by this, as they try to squeeze every available raw byte out of the disk.

Deviation is expressed in ppm. Positive values mean a higher raw capacity than normal, and negative values mean a lower raw capacity than normal. The deviation can be measured using the `floppymeter` program.

[3] drives do not always rotate at exactly 5 or 6 rotations per second, but some may be slightly faster or slightly slower than spec

`rpm=`*`rotation_speed`*

Prescribed rotation speed of the drive, expressed in rotations per minute. This is 360 for 5 1/4 HD drives, and 300 for all other commonly available drive types. Usually, you do not need to specify this parameter, as it can be derived from the drive's CMOS code. It is useful however for single density drives or other drives not commonly found on a PC. Usually, you do not to specify this parameter, as it can be derived from the drive's form factor and maximal density.

`tpi=`*`cylinder_density`*

This parameter is only meaningful for 5 1/4 drives. It expresses whether the drive is able to use 80 tracks (`tpi=96`) or only 40 (`tpi=48`). Usually, you do not to specify this parameter, as it can be derived from the drive's maximal density: quad density and high density drives are 96 tpi, whereas double density drives are 48 tpi.

5.2 The drive definition file in '`/etc/driveprm`'

'`/etc/driveprm`'[4] contains a dictionary of commonly used media descriptions. Each description is identified by a name, which can then be used by setfdprm or superformat to refer to it, instead of an explicit description.

Each definition starts with `"drive`*`number`*`":`, followed by the actual description. Definitions may be spread over several lines, for better readability. The file may contain comments, which start with # and stop at the end of the line.

6 Storing more data on a floppy disk

This section describes the techniques that are used by Linux' floppy driver and superformat to store more data than usual on a floppy disk.

Each section contains a description of the technique used, lists the usages of the disks formatted using this technique (whether they are bootable, whether they are accessible on MS-DOS and for which kind of filesystems they are suitable) and finishes with a table listing the most interesting formats which can be obtained by the described technique.

The table lists for each format the media type it is used for, the total capacity which can be achieved, the throughput for large reads or writes and the media description for these disks. This description can the be used with superformat to make such disks, or with setfdprm to configure the drive to read/write to them. Some formats (the XDF and XXDF formats) cannot be accessed directly, and thus there is no media description for them. For these, we indicate a formatting command used to make these disks. The formatting command assume that the disk is in the first drive (`/dev/fd0`). Substitute `/dev/fd1` if you want to format XDF or XXDF disks in the second drive.

[4] The actual location of this file depends on the value of the `sysconfdir` compile time configuration variable (see Chapter 11 [Compile-time configuration], page 53 for details)

6.1 More sectors per cylinder

The official formats used by MS-DOS and other operating systems are generally very conservative. It is often possible to fit more sectors on each track than the default by simply reducing the size of the gap between tracks and/or the size of the leftover space at the end of the disk.

For example, a 3 1/2 disk has a raw track capacity of around 12500 bytes. The raw capacity of a floppy disk is not rigorously constant among different boxes, because both the data transfer rate of the floppy controller, and the rotation speed of the drive are subject to small variations. In order to account for these, we have to use a safety margin, and we only use up 12450 bytes of the 12500 bytes that are theoretically available.

A sector contains a header of 62 bytes and 512 bytes of data. A minimum gap of about 45 bytes should be used in order to leave enough time to the floppy controller to "rest" between reading two successive sectors. In total, 619 bytes per sector are thus needed.

This shows that we can fit 12450 / 619 = 20 sectors per track.

Usage: These disks are bootable by Lilo, and can be read in MS-DOS using numerous shareware utilities such as `vgacopy`, or `fdformat` or many others. Check your nearest Simtel mirror.

With dos6, you don't need any add-on utilities, just put the following line in your `config.sys`:

```
drivparm=/d:0 /f:7 /h:2 /s:21 /t:82
             ^    ^ \_____________/
             |    |          |
      drive number |     max geometry
                   |
         drive type, consult the dos help system for details
```

Interesting Formats:

density	tot. cap.	throughput	media description
5 1/4 DD	800KB	30KB/s	dd sect=10
3 1/2 DD	800KB	25KB/s	dd sect=10
3 1/2 HD	1600KB	50KB/s	hd sect=21
3 1/2 ED	3200KB	100KB/s	ed sect=42

The `--dd`, `--hd` and `--ed` options describe the density of the media to be formatted (double density, high density or extra density).

The `-s` options describes the number of 512 byte sectors per track.

6.2 Using interleave

After having read a sector, the floppy controller needs to "rest" for a short time. This time is used to compute checksums, to reset internal circuitry, etc. During this time, the floppy disk continues to rotate, and the "rest" time thusly translates to a certain minimal gap size. If a smaller gap is used, the next sector header flies by the read-write head before the floppy controller is ready again to pick up the data. Thus, it has to wait until the next disk rotation until that sector comes back again. This leads to an unacceptably low

throughput, as now the system can only read one sector per rotation instead of all sectors in one rotation. If we want to use smaller gaps, we have thus to use *sector interleaving*. This technique consists in arranging the sectors in a way such that the next logical sector does not immediately follow the current sector, but instead another sector is inserted between two successive sectors. Instead of having the following order:

```
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21
```

we would use the following order:

```
1,12,2,13,3,14,4,15,5,16,6,17,7,18,8,19,9,20,10,21,11,
```

This new order allows the floppy controller to rest during the whole time that sector 12 flies by between reading sector 1 and 2. This technique still cuts throughput in half, because two rotations are needed (one for reading sectors 1 to 11, and the second to read sectors 12 to 21). However, this is far better than the 21 rotations which would be needed without interleave.

This technique allows us to use a gap size of just 1, and thus fit 21 sectors on one track.

Usage: Once formatted, interleaved disks can be used in a similar way to disks which have simply more tracks. They can be accessed using `vgacopy` in Dos, you can boot from them using Lilo, and you may install any filesystem on them.

Interesting Formats:

density	tot. cap.	throughput	media description
5 1/4 HD	1440KB	27KB/s	hd sect=18
3 1/2 HD	1680KB	26KB/s	hd sect=21
3 1/2 ED	3360KB	52KB/s	ed sect=42

You don't need to tell `superformat` to use interleaving, it figures out by itself when interleaving is needed. You don't need to tell `setfdprm` either that a disk is interleaved, as this information is not needed to read the disk

6.3 Sector skewing

Sector skewing is a technique that allows bigger throughputs. It does not increase the capacity of the disk. Sector skewing is only relevant during formatting. Sector skewed disks are indistinguishable from non-skewed disk by software, except for a different throughput.

The principle of sector skewing is to start each track a little bit later than the previous one, i.e. the first logical sector of the second cylinder would for exemple lie near the sixth logical sector of the first cylinder. This is done in order to account for the time needed to seek the drive head from the first cylinder to the second. Without skewing, the first sector would already have passed the drive head after seeking, and we would need to wait for a whole rotation for it to come back again.

By default, superformat applies appropriate skewing to all formats, and the listed throughput values refer to skewed disks. It is possible to provide different values for the skew using the `--head_skew` and `--track_skew` parameters. `head_skew` refers to the offset between both sides of the same cylinder, and `track_skew` refers to the offset of two consecutive cylinders. 0 means no skew.

6.4 More Cylinders

Many nominally 40-cylinders or 80-cylinder drives are capable of more cylinders, usually 41 and 83 respectively. These can be used to get extra capacity. However, not all drives can seek to these unofficial extra cylinders, and even on drives which can, these extra cylinders tend to be less reliable.

WARNING: Although most drives are able to use 83 cylinders, some may not. If your drive is making strange noises while accessing these tracks.

Although most drives support more than 80 tracks, I have heard rumors that some do not, and repeatedly trying to read beyond track 80 might be damaging to them. In order to know wether your drive supports more than eighty tracks, first set the number of allowed tracks to 82. (using `floppycontrol --cylinders 82 -d` *drive*)

Then format a disk with a 82 track format (for example '`/dev/fd0H1722`'), and copy on or several files to the disk until there are less than 18 KB of free space on the disk. Then eject and reinsert the disk, and compare the files on the disk with the originals. If they are the same, your drive does support 82 tracks. If so, you might want to go further and try with 83 cylinders: '`/dev/fd0H1743`') This single experience should not damage the drive, although repeating it many times may be dangerous.

If you do have a drive which supports more than 80 cylinders, you have to call `floppycontrol --cylinders 82` ***`drive`*** before you can use the extra cylinders. You may put this line into your '`/etc/rc.local`', so that the driver is automatically configured for the addition cylinder after each boot.

If on the other hand your drive doesn't support more than 80 tracks, you should remove the entries for formats 13-19 from your '`/dev`' directory after running `MAKEFLOPPIES`, and you should call `floppycontrol --cylinder 80` ***`drive`*** from your '`/etc/rc.local`' (or `floppycontrol --cylinder 40` ***`drive`*** for 5 1/4 DD drives).

By default, 83 cylinder are enabled for any high density and double density drives. 3 1/2 double density drives have 80 cylinders enables, and 5 1/4 double density drives have 40 enabled.

Usage: These disks can be booted from using LILO, and can be accessed in MS-DOS using `vgacopy`.

Interesting Formats:

All formats presented in the two previous sections may be amended to use 83 cylinders instead of 80. Just add the `cyl=83` to the format description for superformat. Using more cylinders has no effect on the throughput.

density	tot. cap.	throughput	media description
5 1/4 HD	1494KB	27KB/s	hd sect=18 cyl=83
3 1/2 HD	1743KB	26KB/s	hd sect=21 cyl=83
3 1/2 ED	3486KB	52KB/s	ed sect=42 cyl=83

6.5 Larger sectors

The floppy controller allows us to use larger sectors than the default size of 512 bytes. All powers of two larger than 256 bytes are acceptable sector sizes. Large sectors have the same

header and gap sizes than smaller sectors, thus the overhead per byte of data is smaller. A little calculation shows this: A 1024 byte sector takes up at least 1024+62+1 = 1087 raw bytes. You can fit eleven sectors of this size into a 12450 byte track. This represents 11KB of data per track, versus the 10.5KB only that can be achieved with 512 byte sectors.

Usage: MS-DOS and other operating systems cannot normally read these formats. Lilo is not yet able to boot from this kind of disks.

Performance: When any portion of one of these larger sectors is read, the entire sector must be read. When any portion of such a sector is written to, the entire sector must be read, and then written back with just the necessary portion modified. Both of these circumstances can entail worse performance than are listed in this table for small reads and (especially) small writes.

Interesting Formats:

```
density     tot. cap.    throughput  media description
5 1/4 HD    1440KB        30KB/s     dd sect=9 ssize=1KB
3 1/2 HD    1760KB        55KB/s     hd sect=11 ssize=1KB
3 1/2 ED    3520KB       110KB/s     ed sect=11 ssize=2KB
```

The `ssize` parameter of the format description indicates the sector size to be used.

6.6 Mixed sector size (MSS) formats

Using larger sectors has the disadvantage that the granularity is larger. For example, when using 4096 byte sectors, there is enough space to fit two sectors in a track of 12450 bytes raw capacity, but not three. However, the two sector format leaves plenty of space available (4132 bytes), in which smaller sectors would fit. For example, these 4142 raw bytes can be put to good use by filling them with a 2 KB sector (2048+62), plus an 1 KB sector (1024+62) and a 512 byte sector, leaving still 362 bytes for gaps.

Mixed sector size formats take advantage of this by using sectors of several different sizes on a same track. This way, a maximum capacity of 12KB per track, distributed in one 8k sector and one 4k sector can be achieved.

Usage: There is no known MS-DOS utility which can read basic MSS disks. Lilo is not yet able to boot from this kind of disks.

Performance: As any format with larger sectors, the performance for small reads and writes is worse due to the larger granularity.

Interesting Formats:

```
density     tot. cap.    throughput    media description
3 1/2 DD     880KB        28KB/s       hd tracksize=11b mss
5 1/4 HD    1600KB        30KB/s       hd tracksize=10KB mss
3 1/2 DD     880KB        28KB/s       hd tracksize=11b mss
3 1/2 HD    1840KB        28KB/s       hd tracksize=23b mss
3 1/2 HD    1920KB        30KB/s       hd tracksize=12KB mss
3 1/2 ED    3680KB        56KB/s       ed tracksize=23KB mss
3 1/2 ED    3840KB        60KB/s       ed tracksize=24KB mss
```

For MSS formats, the system figures out the most efficient repartition of sector sizes by itself. You do not need to describe the number of sectors and their size. For MSS disks, the capacity of one track is described directly, using the `tracksize` parameter.

The `1920KB` and `3840KB` formats may be unreliable on some computers.

6.7 Smart use of the data transfer rate

Due to different drive rotations speeds, 5 1/4 double density disks and 3 1/2 double density disks are accessed using different raw data transfer rates (300 kb/s for the faster spinning 5 1/4 disks, and only 250 kb/s for the slower spinning 3 1/2 disks). The method described in this section consists in using the faster data transfer rate intended for 5 1/4 disks on 3 1/2 disks, and thus boost the raw capacity per track of these disks. This is possible because 300 kb/s is still low enough not to excede the specification of the disk surface of a double density disk (which 500 kb/s would).

Usage: this method is only available for 3 1/2 double density disks. The disk obtained cannot be booted from by LILO, and are inaccessible from MS-DOS.

The following table shows the media description for a format using this method in conjuncion with the previous methods:

```
density     tot. cap.    throughput    media description
3 1/2 DD     1120KB         17KB/s     qd tracksize=7KB mss
```

We use the `QD` density selector to describe this particular DTR set-up, although the acronym `QD` is already taken to name 96tpi double density 5 1/4 disks. However, as this dtr trickery is only meaningful for 3 1/2, we hope that there will be no ambiguity.

6.8 2M formats

2M formats use a standard geometry (18 normal sized sectors) on the first side of the first cylinder, and an MSS geometry on the rest of the disk. They are inspired for Ciriaco Garcia de Celis' 2M utility for MS-DOS.

The advantage of 2M disks over simple MSS disks is twofold:

- They can be accessed from DOS
- They do not need an autodetect entry, as the boot sector is readable using a standard geometry. Mtools can then use the information contained in the boot sector to configure the floppy driver to read the rest of the disk.

Although 2m disk have less sectors on the first track than on the others, the Linux floppy driver, and 2M's low level floppy access routines pretend that it contains the same number of sectors. The missing sectors are called phantom sectors. Writes to these sectors are ignored, and reads return random data. In order to make up for this, 2M and mtools pretend that there is a duplicate FAT in the missing sectors, which is simulated by using data from the first (real) FAT. Thus 2M disks work fine for their intended purpose, which is to hold an MS-DOS filesystem. **Never use 2M disks for anything other than a MS-DOS filesystem. For example, never make an ext2 filesystem on a 2M disk. If you need a high capacity ext2 filesystem (or minix fs, raw tar or cpio archive), use the corresponding MSS format instead**

Usage: 2M disks are not bootable by LILO. They can be accessed in MS-DOS using the 2M utility. 2M can be found at ftp://FTP.Coast.NET/SimTel/msdos/diskutil/2m30.zip or at any other simtel mirror. **2M disks are not suitable for non MS-DOS filesystems**.

Performance: Just as with MSS disks, performance is bad for small reads and writes.

To describe a 2M format, add the keyword `2m` to its media description:

```
density     tot. cap.    throughput    media description
3 1/2 HD    1840KB        28KB/s       hd tracksize=23b mss 2m
```

6.9 XDF formats

XDF is the format used for OS/2 installation floppies.

Just like 2M, XDF uses mixed sector sizes on "generic tracks". The first cylinder uses 512 byte sectors.

However, for XDF disks, the logical order of the sectors on a given track, and their physical order is not the same. This allows a faster access, in a similar way that interleaving does for disk with normal sized sectors and too small gaps. XDF's sector arrangement allows it to read sectors alternatively from both sides, i.e. the first sector from side 0, the second from side 1, and the third from side 0 again. This differs from the usualy formats, where first the entire side 0 is read, and then the entire side 1. This technique allows to read both sides of a disk in roughly three rotations.

The following example illustrates how this is done. In our example we use the XDF format used for 3 1/2 HD disks, which contains one 8KB sector, one 2KB sector, one 1KB sector, and one 512 byte sector per track. The upper line represents the sectors on side 0, and the lower line represents the sectors on side 1. Different numbers represent different sectors. Repeated identical numbers represent a single larger sector. As the disk is circular, some sectors wrap around at the end: we find parts of the 8KB sector, represented by 6, both at the beginning and at the end of each track.

```
position:|          1         2         5         4
         | 12345678901234567890123456789012345678901
         |===========================================
side 0:  | 66333322444444466666666666666666666666666
side 1:  | 66664444444223333666666666666666666666666

2   512 byte sector
3   1KB sector
4   2KB sector
6   8KB sector
```

When reading a track, sectors are read in the following order:

```
sector id    head    position at start     position at end
-------------------------------------------------------------
    3          0              3                  7
    4          0              9                 16
    6          1             18                  5    (1st wrap around)
    2          0              7                  9
    2          1             12                 14
    6          0             16                  3    (2nd wrap around)
    4          1              5                 12
```

```
3          1              14                 18
```

We notice that the start of each sector happens at least 2 units of position (around 300 bytes), after the end of the previously read sector, thus allowing the floppy disk controller sufficient time to rest. Moreover, we notice two wrap-arounds, yielding three rotations to read the whole cylinder (the third rotation is due to the fact that we stop at a higher position than we started, and that we also need to allow some time for seeking to the next track).

MSS or 2M formats of the same capacity nead at least 2 rotations per side (i.e. 4 per track), yielding a lower throughput.

Usage: XDF disks are not bootable by LILO. They can be accessed from MS-DOS and OS/2 using `xdfcopy.exe` or `xdf.com`. They are only suitable for MS-DOS filesystems. The floppy driver has no direct support for this format yet, but `mtools` is able to read them using the `FDRAWCMD` ioctl.

Interesting Formats:

density	tot. cap.	throughput	formatting command
5 1/4 HD	1600KB	46KB/s	xdfcopy -0 /dev/fd0
3 1/2 HD	1840KB	38KB/s	xdfcopy -1 /dev/fd0
3 1/2 ED	3840KB	102KB/s	xdfcopy -2 /dev/fd0

The options `-1`, `-2` and `-3` descibre one out of the five formats understoood by `xdfcopy` (3 XDF formats and 2 XDF formats).

6.10 XXDF formats

These use the simple principle as XDF, but use a higher geometry. No new principle is used, these formats are simply more daring (smaller gaps, and smaller margin at the end of the sector).

Usage: XXDF disks are not bootable by LILO, and can't be accessed by MS-DOS. They are only suitable for MS-DOS filesystems. The floppy driver has no direct support for this format yet, but `mtools` is able to read them using the `FDRAWCMD` ioctl. Due to their smaller tolerances, XXDF formats may not work on all drives. Problems may also occur if you write to XXDF disks using a different drive than the one you used to format the disk.

Interesting Formats:

density	tot. cap.	throughput	formatting command
3 1/2 HD	1920KB	45KB/s	xdfcopy -3 /dev/fd0
3 1/2 ED	3840KB	90KB/s	xdfcopy -4 /dev/fd0

7 How autodetection works

The principle of autodetection is rather simple. When a floppy disk is first accessed, and its geometry is not yet known, the floppy driver tries out a list of up to 8 geometries (format descriptions) until one is found that matches (i.e. that makes it possible to read the first sector or track). This list of geometries is called the *autodetection list*. There is one autodetection list per drive type (as indicated in the cmos).

The autodetection list doesn't contain the geometry descriptions themselves, but rather references to entries in the *geometry list* (see Section 3.3 [geometry list], page 6). Each list may contain up to 8 such references. Each reference can be tagged with a `t` flag. If this tag is set, the floppy driver tries to read the whole track when trying out that description; if it is not set, it only tries to read the boot sector.

Reading the whole track is useful to distinguish among geometries which differ only in the amount of sectors per track. In order to do this, put the geometry with the most sectors first in the list, and set its `t` tag. Use the `t` tag only in this case, as it makes autodetection slower.

Autodetection cannot distinguish between geometries that only differ in the number of heads or in the number of tracks.

Autodetection is meant to supply only a first approximation of the actual format of the disk. It supplies enough information to enable a program such as `mtools` to read the boot sector, which contains the exact information. `Mtools` then uses the information contained in the boot sector to set the exact geometry.

The autodetection list is set using the following command:

```
floppycontrol --autodetect list
```

7.1 Example

The following example restores the default autodetection sequence for a 3 1/2 ED drive:

```
floppycontrol --autodetect 7,8,4,25,28,22,31,21
```

The following example changes this sequence, so as to add the 1680KB format (number 11). As only 8 formats are allowed in the autodetection list, we have to dump one entry (we chose the last, which is numbered 21). The 1680KB format is identical with the default 1440KB format except for the number of sectors. Thus we must read the whole track in order to distinguish it from the 18 sector format (`t` flag). Furthermore, the the 1680KB sector format should be detected first, as an 21 sector disk would also matches the standard format with its 18 sectors.

```
floppycontrol --autodetect 11t,7,8,4,25,28,22,31
```

The following example attempts to autodetect CBM 1581 disks along with the more usual formats. CBM 1581 disks are not among the predefined formats. Thus we first have to pick one of the predefined formats and change it so it fits our needs. We may for example pick one of the rarely used 5 1/4 formats, such as h880, which bears number 20). We first make a device node bearing the requested number (so that we have a filename to pass to setfdprm), then we chmod it so it becomes accessible to mortal users, finally we configure the geometry of the new node, and enter it into the autodetection list. We place it at the 4th position, just behind the usual ED, HD and DD formats, and before the more exotic extended formats. Indeed, formats which are nearer to the head of the list are autodetected faster, and hence more commonly used formats should be put nearer to the beginning[5].

[5] except of course if several formats only differ in the number of sectors per track, in which case the formats with the most sectors should come first

```
mknod /dev/fd0cbm1581 b 2 80
chmod 666 /dev/fd0cbm1581
setfdprm /dev/fd0cbm1581 DD DS sect=10 cyl=80 ssize=512 fmt_gap=35 gap=12 swapsides
floppycontrol --autodetect 7,8,4,20,25,28,22,31
```

Some formats use more than 80 tracks. It is not possible for the kernel to autodetect the number of tracks in a reasonable time, so you have to use a sufficiently recent version of mtools to set the number of tracks according to the boot sector of the disk. Mtools 3.0 and up are ok. This doesn't obviously work with disks containing something else than a MS-DOS filesystem.

8 Configuring the floppy driver via lilo or insmod

The floppy driver is configured using the `floppy=` options in lilo. These options can be typed at the boot prompt, or entered in the lilo configuration file.

Example: If your kernel is called `linux-2.0`, type the following line at the lilo boot prompt (if you have a thinkpad):

```
linux-2.0 floppy=thinkpad
```

You may also enter the following line in '`/etc/lilo.conf`', in the description of `linux-2.0`:

```
append = "floppy=thinkpad"
```

Several floppy related options may be given, example:

```
linux-2.0 floppy=daring floppy=two_fdc
append = "floppy=daring floppy=two_fdc"
```

If you give options both in the lilo config file and on the boot prompt, the option strings of both places are concatenated, the boot prompt options coming last. That's why there are also options to restore the default behaviour.

If you use the floppy driver as a module, use the following syntax: `insmod floppy 'floppy="`*options*`"'`. (This line may be unreadable in the info version of this document. If so, please refer to the printed version).

Example:

```
insmod floppy 'floppy="daring two_fdc"'
```

Note that in this case `floppy=` should only be typed out once, and not once for each option. You need at least modules-1.3.57 for this method. However, the older environment variable based syntax is still available:

Bourne (sh, bash, ksh, zsh) syntax:

```
floppy="daring two_fdc" insmod floppy
```

C-shell (csh, tcsh) syntax:

```
setenv floppy "daring two_fdc" ; insmod floppy
```

Some versions of insmod are buggy in one way or another. If you have any problems (options not being passed correctly, segfaults during insmod), first check whether there is a more recent version. If there isn't, use the old method using environment variables. Problems with `insmod` happen mostly for options involving both a number and a string,

such as `floppy=0,4,cmos`. Options only involving strings, such as `floppy=daring` are not affected.

The floppy related options include:

`floppy=daring`
: Tells the floppy driver that you have a well behaved floppy controller. This allows more efficient and smoother operation, but may fail on certain controllers.

`floppy=0,daring`
: Tells the floppy driver that your floppy controller should be used with caution.

`floppy=one_fdc`
: Tells the floppy driver that you have only floppy controller (default)

`floppy=two_fdc`
`floppy=`*address*`,two_fdc`
: Tells the floppy driver that you have two floppy controllers. The second floppy controller is assumed to be at *address*. If *address* is not given, 0x370 is assumed. two_fdc is implied if you use the cmos option with a drive of id 4 to 7.

`floppy=thinkpad`
: Tells the floppy driver that you have a Thinkpad. Thinkpads use an inverted convention for the disk change line.

`floppy=0,thinkpad`
: Tells the floppy driver that you don't have a Thinkpad.

`floppy=omnibook`
`floppy=nodma`
: Tells the floppy driver not to use Dma for data transfers. This is needed for instance on some HP Omnibooks, which don't have a workable DMA channel for the floppy driver. This option is also useful if you frequently get "Unable to allocate DMA memory" messages. Indeed, dma memory needs to be continuous in physical memory, and is thus harder to find, whereas non-dma buffers may be allocated in virtual memory. However, I advise against this if you have an FDC without a FIFO (8272A or 82072). 82072A and later are OK. You also need at least a 486 to use nodma. If you use nodma mode, I suggest you also set the FIFO threshold to 10 or lower, in order to limit the number of data transfer interrupts.

`floppy=dma`
: Tells the floppy driver that a workable DMA channel is available (the default).

`floppy=nofifo`
: Disables the FIFO entirely. This is needed if you get "Bus master arbitration error" messages from your ethernet card (or from other devices) while accessing the floppy.

`floppy=fifo`
: Enables the FIFO (default)

`floppy=`*threshold*`,fifo_depth`
: Sets the FIFO threshold. This is mostly relevant in DMA mode. If this is higher, the floppy driver tolerates more interrupt latency, but it triggers more

interrupts (i.e. it imposes more load on the rest of the system). If this is lower, the interrupt latency should be lower too (faster processor). The benefit of a lower threshold is less interrupts.

To tune the fifo threshold, switch on over/underrun messages using `floppycontrol --messages`. Then access a floppy disk. If you get a huge amount of `Over/Underrun - retrying` messages, then the fifo threshold is too low. Try with a higher value, until you only get an occasional Over/Underrun. It is a good idea to compile the floppy driver as a module when doing this tuning. Indeed, it allows to try different fifo values without rebooting the machine for each test. Note that you need to do `floppycontrol --messages` every time you re-insert the module.

Usually, tuning the fifo threshold should not be needed, as the default (0xa) is reasonable.

`floppy=`*`drive`*`,`*`type`*`,cmos`

Sets the cmos type of *drive* to *type*. Additionnaly, this drive is allowed in the bitmask. This is useful if you have more than two floppy drives (only two can be described in the physical cmos), or if your BIOS uses non-standard CMOS types. The CMOS types are:

`0`	unknown or not installed
`1`	5 1/4 DD
`2`	5 1/4 HD
`3`	3 1/2 DD
`4`	3 1/2 HD
`5`	3 1/2 ED
`6`	3 1/2 ED

Note that there are two valid types for ED drives. This is because 5 was initially chosen to represent floppy tapes, and 6 for ED drives. AMI ignored this, and used 5 for ED drives. That's why the floppy driver handles both) Setting the CMOS to 0 for the first two drives (default) makes the floppy driver read the physical cmos for those drives.

`floppy=unexpected_interrupts`

Print a warning message when an unexpected interrupt is received (default behaviour)

`floppy=no_unexpected_interrupts`
`floppy=L40SX`

Don't print a message when an unexpected interrupt is received. This is needed on IBM L40SX laptops in certain video modes. (There seems to be an interaction between video and floppy. The unexpected interrupts only affect performance, and can safely be ignored.)

`floppy=broken_dcl`

Don't use the disk change line, but assume that the disk was changed whenever the device node is reopened. Needed on some boxes where the disk change

line is broken or unsupported. This should be regarded as a stopgap measure, indeed it makes floppy operation less efficient due to unneeded cache flushings, and slightly more unreliable. Please verify your cable, connection and jumper settings if you have any DCL problems. However, some older drives, and also some Laptops are known not to have a DCL.

`floppy=debug`
: Print debugging messages

`floppy=messages`
: Print informational messages for some operations (disk change notifications, warnings about over and underruns, and about autodetection)

`floppy=silent_dcl_clear`
: Uses a less noisy way to clear the disk change line (which doesn't involve seeks). Implied by daring.

(There are other options as well, but they are considered obsolete, and thus they are not documented here)

9 Floppy ioctls

All these ioctl's may be issued using the floppycontrol program. (See also floppycontrols man page)

`FDSETPRM` sets the geometry (number of tracks, heads and sectors, etc) of a drive.

`FDDEFPRM` sets the geometry in a permanent way (not cleared after a disk change)

`FDGETPRM` read a previously set drive geometry (or an autodetected geometry) back.

`FDCLRPRM` makes the driver forget the geometry for a given drive (to trigger autodetection)

`FDFLUSH` forgets the contents of the floppy buffers. CAUTION: This doesn't write dirty buffers to the disk. Use fsync first.

`FDGETDRVTYP`
: displays the type of a drive (name parameter). This is used by `MAKEFLOPPIES`. For the naming convention, see the description of the `MAKEFLOPPIES` script. For formats which work in several drive types, `FDGETDRVTYP` return a name which is appropriate for the oldes drive type which supports this format.

`FDSETDRVPRM`
: sets various drive parameters.

`FDGETDRVPRM`
: reads these parameters back.

`FDGETDRVSTAT`
: gets the cached drive state (disk changed, write protected et al.)

`FDPOLLDRVSTAT`
: polls the drive and return its state.

`FDGETFDCSTAT`
: gets the floppy controller state.

`FDRESET` resets the floppy controller under certain conditions.

`FDRAWCMD` sends a raw command to the floppy controller.

`FDWERRORCLR`
: clear the write error stats.

`FDWERRORGET`
: gets the write error stats.

`FDSETMAXERRS`
: sets the error thresholds (when to display error messages on the console, and when to abort operations). The maxerror structure is part of the drive parameters, but this ioctl is needed in addition to `FDSETDRVPRM` because `FDSETDRVPRM` is only accessible to the superuser whereas `FDSETMAXERRS` is accessible to whoever has write access to the floppy device.

`FDMSGON/FDMSGOFF`
: switch informational messages on/off. This flag is part of the drive parameters as well, but `FDMSGON/FDMSGOFF` don't need superuser status.

There are other ioctls as well, but they are considered obsolete and their use is discouraged.

10 Command list

This section describes the available fdutils commands, and the command line parameters that each of them accepts.

10.1 diskd

The diskd command has the following syntax:

```
diskd [-d drive] [-i interval] [-e command]
```

Diskd waits for a disk to be inserted into a given *drive*, and then either executes the *command* or exits. This program can be used to automatically mount a disk as soon as it is inserted.

10.1.1 Warning

This program works by switching the motor on for a very short interval, and then seeking to track -1. This might damage hardware in the long run. Amigas, which also use these techniques, are known for having problems with their disk drives no longer spinning up properly after a few month of usage.

10.1.2 Options

`-d drive` Selects the drive to observe for disk insertion. By default, drive 0 (`/dev/fd0`) is observed.

`-i interval`
Selects the polling interval. The interval is given in tenths of seconds. Default is 10 (one second).

`-e command`
Gives the command to be executed when a disk is inserted. If no command is given the program simply exits. Typically, the command mounts the disk. It can be a shell scripts which probes for several filesystems and disk geometries until it succeeds.

10.1.3 Bugs

- Automatic unmounting cannot yet be handled. It is indeed not enough to scan for disk removal, because when the disk is removed, it is already too late: There might be some buffers needing flushing. However, the `fdmountd` program allows automatic unmounting by using the `SYNC` mount options, which switches off write buffering (see Section 10.3 [fdmount], page 28).
- The drive motor is running all the time, and on some computers, the drive led flickers at each time the drive is polled.

10.2 diskseekd

Several people have noticed that Linux has a bad tendency of killing floppy drives. These failures remained completely mysterious, until somebody noticed that they were due to huge layers of dust accumulating in the floppy drives. This cannot happen under Messy Dos, because this excuse for an operating system is so unstable that it crashes roughly every 20 minutes (actually less if you are running Windows). When rebooting, the BIOS seeks the drive, and by doing this, it shakes the dust out of the drive mechanism. `diskseekd` simulates this effect by seeking the drive periodically. If it is called as `diskseek`, the drive is seeked only once.

10.2.1 Options

The syntax for `diskseekd` is as follows:

```
diskseekd [-d drive] [-i interval] [-p pidfile]
```

`-d drive` Selects the drive to seek. By default, drive 0 (‘`/dev/fd0`’) is seeked.

`-i interval`
Selects the cleaning interval, in seconds. If the interval is 0, a single seek is done. This is useful when calling diskseek from a crontab. The default is 1000 seconds (about 16 minutes) for `diskseekd` and 0 for `diskseek`.

`-p` *pidfile*
: Stores the process id of the diskseekd daemon into *pidfile* instead of the default '`/var/run/diskseekd.pid`'.

10.2.2 Bugs

1. Other aspects of Messy Dos' flakiness are not simulated.
2. This section lacks a few smileys.

10.3 fdmount

```
fdmount [-l] [--list] [-d] [--daemon] [--detach]
[-i interval] [--interval interval] [-o mount-options]
[-r] [-readonly] [-s] [--sync] [--nosync] [--nodev]
[--nosuid] [--noexec] [-f] [--force] [-h] [--help]
[drivename] [mountpoint]

fdumount [-f] [--force] [drivename]

fdlist

fdmountd [-i interval] [--interval interval] [-r]
[-readonly] [-s] [--sync] [--nosync] [--nodev]
[--nosuid] [--noexec] [--help] [drivename] [mountpoint]]
```

The `fdmount` program mounts a floppy disk in the specified drive. It tries to figure out the exact format and filesystem type of the disk from data in the disk's boot sector or super block and the auto-detected track layout.

Currently, fdmount supports the filesystems `minix`, `ext`, `ext2`, `xia`, and `msdos`, and includes special support for disks formatted by the `2M` utility for MS-DOS.

It also checks whether the disk is write protected, in which case it is mounted read-only.

The symbolic *drivename* is (currently) one of '`fd[0-7]`', corresponding to the special device files '`/dev/fd[0-7]`'. If *drivename* is not specified, '`fd0`' is assumed.

The disk is mounted on the directory *mountpoint*, if specified, or on '`/fd[0-7]`'. In either case, the mount point must be an existing, writable directory.

Due to a bug in the floppy driver (?), the polling interval (-i flag) must be longer than the spindown offset. Thus you need to do (for example) floppycontrol –spindown 99 before starting fdmountd in daemon mode

10.3.1 Options

`-l --list`
: List all known drives with their symbolic name, type, and mount status.

`-d --daemon`
: Run in daemon mode (see below).

`--detach` Runs daemon in background, and detaches it from its tty. Messages produced after the fork are logged to syslog.

`-p` *file*
`--pidfile` *file*
Dumps the process id of the daemon to *file*. This makes killing the daemon easier: `kill -9 ‘cat` *file*`‘`

`-i` *interval*
`--interval` *interval*
Set the polling interval for daemon mode. The unit for *interval* is 0.1 seconds, the default value is 10 (i.e. 1 second).

`-o` *options*
`--options` *options*
Sets filesystem-specific options. So far, these are only available for DOS and Ext2 disks. The following DOS options are supported: `check`, `conv`, `dotsOK`, `debug`, `fat`, `quiet`, `blocksize`. The following Ext2 options are supported: `check`, `errors`, `grpid`, `bsdgroups`, `nogrpid`, `sysvgroups`, `bsddf`, `minixdf`, `resgid`, `debug`, `nocheck`. When running as a daemon, options not applying to the disk that is inserted (because of its filesystem type) are not passed to mount.

`-r --`*readonly*
Mount the disk read-only. This is automatically assumed if the disk is write protected.

`-s --`*sync*
Mount with the `SYNC` option.

`--nosync` Mounts without the `SYNC` option, even when running as daemon.

`--nodev` Mount with the `NODEV` option. Ignored for `msdos` filesystems, otherwise always set for non-root users.

`--nosuid` Mount with the `NOSUID` option. Ignored for `msdos` filesystems, otherwise always set for non-root users.

`--noexec` Mount with the `NOEXEC` option.

`-f --`*force*
Attempt a mount or unmount operation even ‘`/etc/mtab`’ says that the drive is already mounted, or not mounted, respectively. This option is useful if ‘`/etc/mtab`’ got out of sync with the actual state for some reason.

`-h --`*help*
Show short parameter description

10.3.2 Security

When mounting on the default mount point, the mount points' owner is set to the current user, and the access flags according to the user's umask. For a specified mountpoint, owner and permissions are left unchanged. Default mount points are called `/fd0`, `/fd1`, . . . , `/fd7`.

The user running fdmount must have read access to the floppy device for read only mounts, and read/write access for read/write mounts.

Fdmount can be run suid root, allowing users to mount floppy disks. The following restrictions are placed upon non-root users:

- If a mountpoint is specified explicitly, it must be owned by the user.
- A user may only unmount a disk if the mount point is owned by the user, or if it the disk has been mounted by the same user.
- Non-msdos disks are automatically mounted with the `nodev` and `nosuid` flags set.

However, **do not rely on fdmount being secure at the moment**.

10.3.3 Daemon mode

In daemon mode, the specified drive is periodically checked and if a disk is inserted, it is automatically mounted.

When the disk is removed, it is automatically unmounted. However, it is recommended to unmount the disk manually *before* removing it. In order to limit corruption, disks are mounted with the SYNC option when running in daemon mode, unless the `--nosync` flag is given.

Note that this mode has some potential drawbacks:

- Some floppy drives have to move the drive head physically in order to reset the disk change signal. It is strongly recommended not to use daemon mode with these drives. See Section 10.5 [floppycontrol], page 36, for details.
- If a disk does not contain a filesystem (e.g. a tar archive), the mount attempt may slow down initial access.
- As fdmount cannot identify the user trying to use the disk drive, there is no way to protect privacy. Disks are always mounted with public access permissions set.

10.3.4 Diagnostics

`error opening device` *`name`*
`error reading boot/super block`
: fdmount failed to read the first 1K of the disk. The disk might be damaged, unformatted, or it may have a format wich is unsupported by the FDC or the Linux kernel.

`unknown filesystem type`
: No magic number of any of the supported filesystems (see above) could be identified.

`sorry, can't figure out format (`*`fs`* `filesystem)`
: The size of the filesystem on the disk is incompatible with the track layout detected by the kernel and an integer number of tracks. This may occur if the filesystem uses only part of the disk, or the track layout was detected incorrectly by the kernel.

`failed to mount` *fs*`> <`*size*`K-disk`
: The actual `mount` system call failed.

`failed to unmount`
: The actual `unmount` system call failed.

`cannot create lock file /etc/mtab~`
: If '`/etc/mtab~`' exists, you should probably delete it. Otherwise, check permissions.

`Can't access` *mountpoint*
: Most probably, the default or specified mount point does not exist. Use mkdir.

mountpoint `is not a directory`
: The mountpoint is not a directory.

`not owner of` *mountpoint*
: Non-root users must own the directory specified as mount point. (This does not apply for the default mount points, /fd[0-3].)

`No write permission to` *mountpoint*
: Non-root users must have write permission on the mount point directory.

`Not owner of mounted directory: UID=`*uid*
: Non-root users cannot unmount if the mount point is owned (i.e. the disk was mounted) by another user.

`invalid drive name`
: Valid drive names are '`fd0`', '`fd1`', etc.

`drive` *name* `does not exist`
: The drive does not exist physically, is unknown to the Linux kernel, or is an unknown type.

`Drive` *name* `is mounted already`
: Trying to mount a drive which appears to be mounted already. Use the `--force` option if you think this is wrong.

`Drive` *name* `is not mounted`
: Trying to unmount a drive which does not appear to be mounted. Use the `--force` option if you think this is wrong.

`ioctl(...) failed`
: If this occurs with the `FDGETDRVTYP` or `FDGETDRVSTAT`, ioctl's you should probably update your Linux kernel.

`mounted` *fs size*`-disk (`*options*`)`
: Success message.

10.3.5 Bugs

- Fdmount should be more flexible about drive names and default mount points (currently hard coded).

- Probably not very secure yet (when running suid root). Untested with ext and xia filesystems.
- Can't specify filesystem type and disk layout explicitly.
- In daemon mode, the drive light stays on all the time.
- Some newer filesystem types, such as vfat are not yet supported.

10.4 fdrawcmd

```
fdrawcmd [drive=drive] [rate=rate]
[length=length] [repeat=repeat]
[cylinder=physical-cyl] command [paramters ...] [mode]
```

`fdrawcmd` is used to send raw commands to the floppy disk controller, after having selected a given drive. You must have write permission to the selected drive.

When writing to a disk, data is read from stdin; when reading, data is printed to stdout. Diagnostic messages, return values from the controller, and the value of the disk change line after the command are printed to stderr.

10.4.1 Options

All numbers may be given in octal (0211), decimal (137), or hexadecimal (0x89).

`drive=`*`drive`*
: Selects the drive. The default is drive 0 ('`/dev/fd0`').

`rate=`*`rate`*
: Selects the data transfer rate. Use 0 for high density disks, 1 for double density 5 1/4 disks (or 2 Mbps tapes, if the appropriate rate table is selected), and 2 for double density 3 1/2 disks.

`length=`*`length`*
: Describes the length of the transferred data for commands reading from and writing to the disk. The default is to continue until end of file.

`repeat=`*`count`*
: Repeat the command *count* times. This only works correctly for commands which don't do any data transfer.

`cylinder=`*`count`*
: Seek to the given cylinder before executing the command

`command`
: The name of the command to send. *command* may be a spelled out name (like `read` or `write`), or a number representing the commands floppy disk controller opcode. A named command has already a mode associated with it, whereas for a number the mode parameter should be described using the `mode` option.

`parameters`
: The parameters for the command (optional, not all commands need parameters).

`mode`
: Various flags or'ed together describing the properties of the command.

10.4.2 Commands

The description of the various floppy commands given in this manpage is very sketchy. For more details get the 82078 spec sheet which can be found at:

`http://www-techdoc.intel.com/docs/periph/fd_contr/datasheets/`

Look for the chapter `COMMAND SET/DESCRIPTIONS`. Older FDCs only support a subset of the commands described therein, but the syntax for the commands that do exist is the same.

10.4.2.1 Commands available on all FDCs

`read` *`drvsel cyl head sect szcod spt rw-gap szcod2`*
: Reads *length* bytes of data from the disk. *drvsel* is the drive selector. Bit 0 and 1 describe the drive, and bit 2 describes the head. The remaining parameters give the cylinder, head (yes, again), sector, size of the sector (128 * 2 ^ *szcod*), sectors per track (`spt`, this is used to switch to the second head when the first side has been read), and size of the read-write gap. *szcod2* should be 0xff. `read` returns *ST0 ST1 ST2* and *cyl head sect szcod* of the next sector to be read; see '`/usr/include/linux/fdreg.h`' .

  N.B. Certain newer floppy disk controllers are buggy, and do not correctly recognize the end of transfer when operating in virtual DMA mode. For these, you need to set `spt` to the id of the last sector to be read (for example, if you intend to read sectors 2, 3, 4, set `spt` to 4, even if the disk has more sectors), and set the `no-mt` flag.

`write` *`drvsel cyl head sect szcod spt rw-gap szcod2`*
: Analogous to `read`.

`sense` *`drvsel`*
: Returns the third status byte (*ST3*)

`recalibrate` *`drvsel`*
: Recalibrates the drive and returns *ST0 ST1*.

`seek` *`drvsel cyl`*
: Moves the head to *cyl* and returns *ST0 ST1*.

`specify` *`drvsel spec1 spec2`*
: Specify various parameters to the drive.

`format` *`drvsel szcod sect-per-track fmt-gap fmt-fill`*
: Formats the cylinder. The new sectors are filled with *fmt-fill*. The header information comes from the input, which is made up of *cyl head sect szcod* quadruples. The *szcod* parameter from the command line is used to describe the actual size of the sectors, and the *szcod* from the input is used to write into the header. However, the first write to these sectors will use the header information, and might overwrite the following sectors if the *szcod* parameter from the command line was too small.

`readid` *drvsel*
: reads the first sector header that comes and returns *ST0 ST1 ST2* and *cyl head sect szcod* of the encountered header.

10.4.2.2 Commands available on 82072 and later

`dumpregs` Prints the contents of the FDCs registers, if supported.

10.4.2.3 Commands available on 82072A and later

`configure` *conf1 conf2 conf3*
: Configures FIFO operation.

10.4.2.4 Commands available on 82077 and later

`version` Echoes 0x90 if the FDC is more recent than 82072A, and 0x80 otherwise

`perpendicular` *rate*
: Sets the perpendicular mode. Use 0 for normal, 2 for 500kb/s perpendicular, and 3 for 1 Mb/s perpendicular.

`seek-out` *drvsel n*
: does a relative seek of *n* cylinders towards cylinder 0.

`seek-in` *drvsel n*
: does a relative seek of *n* cylinders away from cylinder 0.

10.4.2.5 Commands available on 82077AA and later

`lock` Locks the FIFO configuration, so that it survives a FDC software reset.

`unlock` Unlock the FIFO configuration

10.4.2.6 Commands available on 82078

`partid` echoes a byte describing the type of the FDC in the 3 high bits, and the stepping in the three low bits.

`powerdown` *powerconf*
: configures automatic power down of the FDC. The old configuration is echoed

`option` *iso*
: enables/disables ISO formats. Odd values of *iso* enable these formats, whereas even values disable them. ISO formats don't have index headers, and thus allow to fit slightly more data on a disk.

`save` prints out 16 internal registers of the FDC.

`restore` *r1 r2 r3 ... r16*
: restores the 16 internal registers of the FDC.

`format_n_write` *drvsel szcod sect-per-track fmt-gap fmt-fill*
: formats the cylinder and writes initial data to it. The input data is made up of a sequence of headers (4 bytes) and data: *header1 data1 header2 data2 ... headern datan*

`drivespec` *dspec1 dspec2 ... specn terminator*
: chooses rate tables for various drives. Each dspec byte describes one drive. Bits 0 and 1 say which drive is described. Bits 2 and 3 describe the rate table. Only tables 0 and 2 are interesting. Both tables only differ in the meaning og rate 1. For table 0 (the default) rate 0 is 300 kb/s (used for 5 1/4 DD disks), whereas for table 1 it is 2 Mbps (used for fast floppy tape drives). Bit 4 is the precompensation table select bit. It should be set to 0. Bit 5-7 should be zero as well. The *terminator* byte ends the `drivespec` command. It is either 0xc0 or 0x80. If it is 0xc0, no result phase follows; if it is 0x80, the current data rate table configuration for the four drives is echoed.

10.4.3 Modes

The mode option is only needed when you describe the command as a numerical value. Some mode names are also valid command names. They are considered as command name if the command name has not yet been given, and as mode name otherwise.

If you give a command name followed by explicit modes, both the implicit flags of the command name, and the explicit modes are or'ed together.

If on the other hand you give a command name preceded by explicit modes, only the explicit modes are or'ed together.

`read` Read data from disk using DMA.

`write` Write data to the disk.

`intr` Wait for an interrupt.

`spin` wait for the disk to spin up

`disk` Aborts the operation if no disk is in the drive. This only works if you also chose a physical cylinder to seek to.

`no-motor` Don't switch on the drive motor while issuing the command

`no-motor-after`
: Switch off the motor immediately after the command returns.

`fm` Uses the FM version of the `read`, `readid`, `write` and `format` commands.

`no-mt` Do not use MT (multitrack) mode for the `read`, `readid` and `write` commands. This is needed on certain broken FDC's which don't recognize end of transfer when running in `nodma` mode. In order to use these safely, set `no-mt`, and chose the id of the last sector to be read as `sect-per-track`.

`fdrawcmd` opens the device node with the `NDELAY` flag. This means that the driver should not try to autodetect the disk type (it might not be formatted), and that it should not reset the FDC. If a reset was needed, the command simply fails. If that happens, execute `floppycontrol --resetnow 0` , and try again.

10.5 floppycontrol

```
floppycontrol [-p] [--pollstate] [--printfdstate]
[-a operation-abort-threshold] [-c read-track-threshold]
[-r recalibrate-threshold] [-R reset-threshold]
[-e reporting-threshold] [-f] [-x] [-d drive][-F] [-T]
[-reset condition] [--debug] [--nodebug] [--messages]
[--nomessages] [--broken_dcl] [--working_dcl] [--inverted_dcl]
[--no_inverted_dcl] [--silent_dcl_clear] [--noisy_dcl_clear]
[-ccmos-type] [-hlt hlt] [-hut hut] [-srt srt] [-o spindown]
[-u spinup] [-s select-delay] [-rps rotations-per-second]
[-O spindown-offset] [-track max-tracks] [-timeout seconds]
[-C check-interval] [-n native-format]
[-autodetect autodetection-sequence] [-P] [--clrwerror]
[--printwerror] [-h]
```

The `floppycontrol` program is used to configure the floppy driver.

10.5.1 General Options

`-h`
`--help` Print a help screen.

`-d` *drive*
`--drive` *drive*
Selects the drive to configure. The default is drive 0 ('`/dev/fd0`').

10.5.2 One time actions

The following `floppycontrol` options don't set a configuration parameter, but perform a one-time action. They are available to anybody who has write access to the drive

`-f`
`--flush` Flushes (throws away) the dirty data buffers associated with this drive.

`-x`
`--eject` Ejects the disk out of the drive (Sparc). The dirty buffers are first committed to disk before ejecting it. Fails if the disk is mounted.

`--reset` *condition*
Resets the FDC under *condition* . Condition may be one of the following:

`0` resets the FDC only if a reset is needed anyways,

`1` resets the FDC also if a raw command has been performed since the last reset, and

`2` resets the FDC unconditionally.

This command may be needed after some failed raw commands (see Section 10.4 [fdrawcmd], page 32).

`-F`
`--formatend`
: Issues an end format ioctl. This might be needed after exiting a `fdformat` in an unclean way. `superformat` is not subject to this.

10.5.3 Printing current settings

`-T`
`--type` Print out the drive name of a floppy device. This is used by the `MAKEFLOPPIES` script. The drive name is a letter (describing the drive type) followed by the capacity of the format in bytes. The letter is E for 3.5 ED drives, H for 3.5 HD drives, D for 3.5 DD drives, h for 5.25 HD drives and d for 5.25 DD drives. The drive type letter corresponds to the oldest drive type supporting the format of this device node (not necessarily the type of the drive refered by this node.) For the generic format nodes (/dev/fd0 et al.) the name of "native format" of the drive is printed, and for the default formats, if a generic format has been redefined, its name becomes `(null)`.

`-p`
`--print` Prints out the configuration of the drive. The names of the various fields are the same as the names of the option to set them, see below.

`-P`
`--printstate`
: Prints out the cached internal state of the driver. The first line lists various attributes about the disk:

  `drive present`
  `disk present`
  `disk writable`
  : These are only updated when the drive is accessed.

  `spinup` is the time when the motor became switched on for the last time.

  `select` is the time when the drive became selected for the last time

  `first_read`
  : is the time when the first read request after the last spin up completed.

  `probed_fmt`
  : is the the index of the autodetected format in the autodetection sequence for this drive.

  `cylinder` is the cylinder where the drive head currently sits. If this number is negative, it has the following meaning:

  - -1 means that the driver doesn't know, but the controller does (a seek command must be issued).
  - -2 means that the controller doesn't know either, but is sure that it not beyond the 80th track. The drive needs a recalibration.

- -3 means that the head may be beyond the 80th track. The drive needs two successive recalibrations, because at each recalibration, the controller only issues 80 move head commands per recalibration.

`maxblock` is the highest block number that has been read.

`maxcylinder`
: is a boolean which is set when a sector that is not on cylinder 0/head 0 has been read. These are used for smart invalidation of the buffer cache on geometry change. The buffer cache of the drive is only invalidated on geometry change when this change actually implies that a block that has already been read changes position. This optimization is useful for mtools which changes the geometry after reading the boot sector.

`generation`
: is roughly the number of disk changes noticed since boot. Disk changes are noticed if the disk is actually changed, or if a flush command is issued and for both cases if any I/O to/from the disk occurs. (i.e. if you insert several disks, but don't do any I/O to them, the generation number stays the same.)

`refs` is number of open file descriptors for this drive. It is always at least one, because floppycontrol's file descriptor is counted too.

`device` is format type (as derived from the minor device number) which is currently being used.

`last_checked`
: is date (in jiffies) when the drive was last checked for a disk change, and a disk was actually in the drive.

`--pollstate`
: Polls the drive and then prints out the internal state of the driver.(`--Printstate` only prints out the cached information without actually polling the drive for a disk change.)

`--printfdcstate`
: Prints out the state of the controller where the target drive is attached to.

`spec1`
`spec2` are the current values of those registers.

`rate` is current data transfer rate

`rawcmd` is true if a raw command has been executed since the last reset. If this is the case, a reset will be triggered when a drive on the same FDC is next opened.

`dor` is the value of the digital output register. The 4 high bits are a bit mask describing which drives are spinning, the 2 low bits describe the selected drive, bit 2 is used to reset the FDC, and bit 3 describes whether this FDC has hold of the interrupt and the DMA. If you have two FDCs, bit 3 is only set on one of them.

`version` is the version of the FDC. See '`linux/include/linux/fdreg.h`' for a listing of the FDC version numbers.

`reset` is true if a reset needs to be issued to the FDC before processing the next request.

`need_configure`
is true if this FDC needs configuration by the `FD_CONFIGURE` command.

`has_fifo` is set if the FDC understands the `FD_CONFIGURE` command.

`perp_mode`
describes the perpendicular mode of this FDC. 0 is non-perpendicular mode, 2 is HD perpendicular mode, 3 is ED perpendicular mode, and 1 is unknown.

`address` is the address of the first I/O port of the FDC. Normally, this is 0x3f0 for the first FDC and 0x370 for the second.

10.5.4 Drive type configuration and autodetection

The following options handle the different available drive types, such as double density vs. high density vs. extra density drives, and 5 1/4 drives vs 3 1/2 drives. Usually the drive type is stored in a non-volatile memory, called CMOS, under the form of an integer ranging from 1 to 6.

Different drive types are able to handle and autodetect different formats (different autodetection lists). They also have different "native format name". The native format is the "usual" format with the highest capacity supported by the drive. (For example 720KB on a double density 3 1/2 drive, and 1.2MB on a high density 5 1/4 drive.)

These settings are only changeable by the super user.

`-c` *`cmos-type`*
`--cmos` *`cmos-type`*
Set the virtual CMOS type of the floppy drive. This is useful if

- the physical CMOS type is wrong (this may happen with BIOSes which use a non-standard mapping),
- you have more than two drives (the physical CMOS may only describe up to two drives).
- you have a BIOS that allows swapping drives A: and B: for DOS.

Right now, this CMOS parameter is not used by the kernel, except for feeding it back to other applications (for instance `superformat`, `floppymeter` or `MAKEFLOPPIES`). It is also possible to supply a virtual CMOS type with the `cmos` boot option (see Chapter 8 [Boottime configuration], page 22). If possible, I recommend you use the boot option, rather than `floppycontrol`, because the boot option also sets any parameters derived from the CMOS type, such as the autodetection list and the native format, whereas `floppycontrol` does not.

`-A` *autodetect-seq*
`--autodetect` *autodetect-seq*

Set the autodetection sequence (see Chapter 7 [Autodetection], page 20) The autodetection sequence is a comma-separated list of at most eight format descriptors. Each format descriptor is a format number optionally followed by the letter `t`. For drive 0, the format number is the minor device number divided by 4. The autodetection sequence is used by the driver to find out the format of a newly inserted disk. The formats are tried one after the other, and the first matching format is retained. To test the format, the driver tries to read the first sector on the first track on the first head when `t` is not given, or the whole first track when `t` is given. Thus, autodetection cannot detect the number of tracks. However, this information is contained in the boot sector, which is now accessible. The boot sector can then be used by mtools to configure the correct number of tracks.

Example:

```
7,4,24t,25
```

means to try out the formats whose minor device numbers are 28 (1.44M), 16 (720KB), 96 (1.76MB), and 100 (1.92MB), in this order. For the 1.76MB format, try to read the whole track at once.

Reading the whole track at once allows you to distinguish between two formats which differ only in the number of sectors. (The format with the most sectors must be tried first.) If you use mtools[6], you do not need this feature, as mtools can figure out the number of sectors without any help from the floppy driver, by looking at the boot sector.

Reading the whole track at once may also speed up the first read by 200 milliseconds. However, if, on the other hand, you try to read a disk which has less sectors than the format, you lose some time.

I suggest that you put the most often used format in the first place (barring other constraints), as each format that is tried out takes 400 milliseconds.

`-n` *native-format*
`--native_format` *native-format*

Set the native format of this drive. The native format of a drive is the highest standard format available for this drive. (Example: For a 5 1/4 HD drive it is the usual 1200K format.) This is format is used to make up the format name for the generic device (which is the name of the native format). This drive name is read back from the kernel by the `MAKEFLOPPIES` script which uses it to decide which device nodes to create.

10.5.5 Configuration of the disk change line

`--broken_dcl`

Assumes that the disk change line of the drive is broken. If this is set, disk changes are assumed to happen whenever the device node is first opened. The physical disk change line is ignored.

[6] Version 3.0 or higher

This option should be used if disk changes are either not detected at all, or if disk changes are detected when the disk was actually not changed. If this option fixes the problem, I'd recommend that you try to trace the root cause of the problem. Indeed, this options results in reduced performance due to spurious cache flushes.

The following hardware problems may lead to a bad disk change line:

- If the floppy cable is not inserted straight, or if it is kinked, the disk change line is likely to suffer, as it is on the edge of the cable. Gently press on both connectors of the cable (drive and controller) to insure that all wires make contact. Visually inspect the cable, and if it shows obvious traces of damage, get a new one.
- On some drives, the locations disk change line may be chosen by jumper. Make sure that your floppy controller and your drive agree on which line is the disk change line.
- Some older drives (mostly double density 5 1/4 drives) don't have a disk change line. In this case, you have no choice other than to leave the `broken_dcl` option on.

`--working_dcl`
: Assumes that the disk change line works all right. Switching from broken to working may lead to unexpected results after the first disk change.

`--inverted_dcl`
: Assumes that this disk drive uses an inverted disk change line. Apparently this is the case for IBM thinkpads.

`--no_inverted_dcl`
: Assumes that this drive follows the standard convention for the disk change line.

`--noisy_dcl_clear`
: Switches off silent disk change line clearing for this drive.

10.5.6 Timing Parameters

This section describes how to configure drive timings. To set these parameters, you need superuser privileges. All times are in "jiffy" units (10 milliseconds), unless otherwise specified.

`--hlt` *`hlt`*
: Set the head load time (in microseconds) for this floppy drive. The head load time describes how long the floppy controller waits after seeking or changing heads before allowing access to a track.

`--hut` *`hut`*
: Set the head unload time (in microseconds) for this floppy drive. The head unload time describes how long the floppy controller waits after an access before directing its attention to the other head, or before seeking.

`--srt` *srt*
: Set the step rate (in microseconds) for this floppy drive. The step rate describes how long the drive head stays on one cylinder when seeking. Setting this value to low (too fast seeks) may make seeks fail, because the motor doesn't follow fast enough.

`-u` *spinup-time*
`--spinup` *spinup-time*
: Set the spinup time of the floppy drive. In order to do read or write to the floppy disk, it must spin. It takes a certain time for the motor to reach enough speed to read or write. This parameter describes this time. The floppy driver doesn't try to access the drive before the spinup time has elapsed. With modern controllers, you may set this time to zero, as the controller itself enforces the right delay.

`-o` *spindown-time*
`--spindown` *spindown-time*
: Set the spindown time of this floppy drive. The motor is not stopped immediately after the operation completes, because there might be more operations following. The spindown time is the time the driver waits before switching off the motor.

`-O` *spindown-offset*
`--spindown_offset` *spindown-offset*
: Set the spindown offset of this floppy drive. This parameter is used to set the position in which the disk stops. This is useful to minimize the next access time. (If the first sector is just near the head at the very moment at which the disk has reached enough speed, you win 200 milliseconds against the most unfavorable situation).

  This is done by clocking the time where the first I/O request completes, and using this time to calculate the current position of the disk.

`-s` *select-delay*
`--select_delay` *select-delay*
: Set the *select delay* of this floppy drive. This is the delay that the driver waits after selecting the drive and issuing the first command to it. For modern controllers/drives, you may set this to zero.

`-C` *check-interval*
`--checkfreq` *check-interval*
: Set the maximal disk change check interval. The disk change line is checked whenever a read or write to the device is issued, and it has not been checked for more than *interval* jiffies.

10.5.7 Debugging messages

This subsection describes how to switch the available debugging messages on and off.

`--debug` Switch debugging output on. The debugging information includes timing information. This option might be useful to fine-tune the timing options for

your local setups. (But for most normal purposes, the default values are good enough.)

`--nodebug`
: Switch debugging output off.

`--messages`
: Print informational messages after autodetection, geometry parameter clearing and dma over/underruns.

`--nomessages`
: Don't print informational messages after these events.

10.5.8 Error Handling Options

The following options configure the behavior of the floppy driver in case of read/write errors. They may be used by any user who has write privileges for the drive. Whenever the floppy driver encounters an error, a retry counter is incremented. If the value of this counter gets bigger than the thresholds described below, the corresponding actions are performed at the next retry. The counter is reset when the read or write finally terminates, whether successfully or not.

`-a` *`operation-abort-trshld`*
`--abort` *`operation-abort-trshld`*
: Tell the floppy driver to stop trying to read/write a sector after *operation-abort-trshld* retries, and signal the I/O error to the user.

`-t` *`read-track-trshld`*
`--readtrack` *`read-track-trshld`*
: Tell the floppy driver to switch from track-reading mode to sector-at-a-time-mode after *read-track-trshld* retries.

`-r` *`recalibrate-trshld`*
`--recalibrate` *`recalibrate-trshld`*
: Tell the floppy driver to recalibrate the drive after *recalibrate-trshld* retries.

`-R` *`reset-treshold`*
`--reset` *`reset-threshold`*
: Tell the floppy driver to reset the controller after *reset-threshold* retries. After a controller reset, the floppy driver also recalibrates all drives connected to that controller.

`-e` *`error-report-trshld`*
`--reporting` *`error-report-trshld`*
: Tell the floppy driver to start printing error messages to the console after *error-report-trshld* retries.

10.5.9 Write error reporting

Due to the buffer cache, write errors cannot always be reported to the writing user program as soon as the write system call returns. Indeed, the actual writing may take place

much later. If a write error is encountered, the floppy driver stores information about it in its per drive write error structure. This write error structure stays until explicitly cleared. It can for example be queried by a backup program which wants to make sure that the data has been written successfully.

`--clrwerror`
: Clears the write error structure.

`--printwerror`
: Prints the contents of the write error structure:

  `write_errors`
  : is a count of how many write errors have occurred since the structure was last cleared.

  `badness`
  : is the maximal number of retries that were needed to complete an operation (reads, writes and formats).

  `first_error_sector`
  : is where the first (chronologically) write error occurred.

  `first_error_generation`
  : is the disk change generation in which did the first write error occurred. The disk change generation is a number which is incremented at each disk change.

  `last_error_sector`
  : and

  `last_error_generation`
  : are similar.

10.5.10 Other drive configuration options

This subsection lists per drive configuration options, which don't fit in any other category. They are available only to the superuser:

`--tracks` *`max-tracks`*
: Set the maximal numbers of physical tracks that this drive may handle. If you have a drive which is only able to handle 80 tracks (making strange noises when you try to format or read a disk with more than 80 tracks), use this option to prevent unprivileged users of damaging your drive by repeatedly reading disks with more than 80 tracks.

  If you trust your users and your disks, you don't need this. With most drives you don't need to worry anyways. See Section 6.4 [More cylinders], page 16, for details.

`-i` *`sector-interleave`*
`--interleave` *`sector-interleave`*
: Set the number of sectors beyond which sector interleaving will be used. This option will only be used by the `FDFMTTRK` ioctl. The `fdformat` command, which is now considered obsolete, uses `FDFMTTRK` ioctl, but `superformat` does not.

10.6 floppymeter

```
floppymeter [-f] [-w warmup-delay] [-W window]
[-c cycles] [-h] drive
```

The `floppymeter` program measures characteristic parameters of a floppy drive and floppy controller, such as the rotation speed of the drive, the data transfer rate of the controller, and the resulting raw capacity of a disk track. To use this program, insert a disposable floppy in the drive, and type `floppymeter` `--`*density*, where density describes the density of the disk used for the test. (Should be one of `dd, hd` or `ed`). **CAUTION: the data on the disk will be erased**. This program should be used to verify whether the drive and controller are out of tolerance if you experience problems with some high capacity formats. It only needs to be run once per drive: although a disk is needed to perform the measurements, the measured data only depend on the drive and the controller, and not on the disk.

To measure the raw capacity of the disk track, the floppymeter program formats the first track of the drive in a special way that allows it to read the raw data (gaps and headers) of the disk. **Thus, all data previously stored on that disk is lost**.

The rotation speed is measured by timing the return time of a `readid` command. In order to gain more precision, the command is issued many times in a row. During this phase, the number of rotations til the start of the test, the average time per rotation til the start, and a sliding average of the times of the last 30 rotations is printed, and updated continously.

The data transfer rate is deduced from the two parameters above.

At the end of the program, all parameters (raw capacity, duration of one rotation, and data transfer rate) are printed again, as well as their relative deviation to the standard value. Finally, it suggests a capacity deviation description line, which can be directly pasted into the drive definition file (See Chapter 5 [Drive descriptions], page 11.).

Usually, the data transfer rate should not deviate more than 150 ppm from the expected value, and the rotation speed of the drive should not deviate more than 3000 ppm from the expected value. If these deviations are bigger, you may get problems with certain high capacity formats.

If the raw capacity of the drive is too small, some high capacity formats may become unformattable on this drive/controller combo.

If on the other hand, the raw capacity of the drive is too big, you may get problems when writing to a disk formatted by this drive on another drive with a smaller raw capacity. In order to avoid this, increase superformats gap parameter (`-G`).

`-h` Prints a short help

`--dd` Tells the program that we use a Double Density disk.

`--hd` Tells the program that we use a High Density disk.

`--ed` Tells the program that we use an Extra Density disk.

`-f` Runs the measurement non interactively. With this option, the program doesn't ask for confirmation, and doesn't display the continously updated values during the rotation speed measurement.

`-W` *Window*

This value describes how many rotations are used for the computation of the sliding average. Default is 30.

`-c` *cycles*

Describes the number of rotations clocked during the rotations speed determination test. Default is 1000.

10.6.1 Bugs

This program is quite new, and may have bugs. Here are a few suggested tests to check its sanity:

- The deviation of the data transfer rate solely depends on the controller. It should not be different between two drives connected to the same controller. However, the drive rotation speed may be different for different drives.
- All data transfer rates (for double, high and extra density) are derived from a same master frequency. Thus the *deviation* of the data transfer rate should be independant of the density used.

10.7 getfdprm

`getfdprm [`*drive*`]`

`getfdprm` prints the current geometry information for *drive* . This information can be set using `setfdprm`

10.8 makefloppies

`MAKEFLOPPIES [-tlvng] [`*drives*`]`

The `MAKEFLOPPIES` shell script creates the new floppy block device node. It uses the floppycontrol program to translate the minor device numbers into meaningful names. It also uses these names to decide whether to create a given block device file or not, depending on the type of the physical drive (for instance, for a 3 1/2 drive, the formats corresponding to a 5 1/4 drive are not created).

If you have more than two floppy drives, you need to tell the kernel the CMOS types of those additional drives using the `floppy=`*drive*`,`*type*`,cmos` lilo option.

If the *drives* parameter is given, only the device nodes for the listed drives are made. By default, all only the two first drives are tried.

`MAKEFLOPPIES` does not work if you redefine your default formats.

Caution: `MAKEFLOPPIES` removes already existing floppy device nodes.

10.8.1 Options

`-t` Use the old naming convention for 3 1/2 devices (e.g. '`fd0H720`' instead of '`fd0u720`').

`-m` Base the name for the created devices on the type of the media (e.g. '`fd0h720`' instead of '`fd0u720`').

`-l` Local. Creates device nodes in the local directory, not /dev

`-v` Verbose

`-n` Dry run. (just report what would be done, do not do anything)

`-g` Group. Allow read/write access to floppy devices only for group '`floppy`'

10.8.2 Bugs

The Makefloppies script does not work on redefined "default" formats, If you redefine default formats, you need to create the needed device nodes manually.

10.9 setfdprm

```
setfdprm [-p] device media-description

setfdprm [-c | -y | -n] device
```

`setfdprm` is a utility that can be used to load disk parameters into the auto-detecting floppy devices and "fixed parameter" floppy devices, to clear old parameter sets and to disable or enable diagnostic messages. These parameters are derived from a media-description, see Chapter 4 [Media description], page 8 for more details.

Without any options, `setfdprm` loads the *device* (for example '`/dev/fd0`' or '`/dev/fd1`') with a new parameter set with the *name* entry found in '`/etc/fdprm`' (usually named 360/360, etc.). For autodetecting floppy devices, these parameters stay in effect until the media is changed. For "fixed parameter" devices, they stay in effect until they are changed again.

`Setfdprm` can also be used by the superuser to redefine the default formats.

10.9.1 Options

`-p` *device name*
: Permanently loads a new parameter set for the specified auto-configuring floppy device for the configuration with *name* in '`/etc/fdprm`'. Alternatively, the parameters can be given directly from the command line.

`-c` *device*
: Clears the parameter set of the specified auto-configuring floppy device.

`-y` *device*
: Enables format detection messages for the specified auto-configuring floppy device.

`-n` *device*
: Disables format detection messages for the specified auto-configuring floppy device.

10.9.2 Bugs

This documentation is grossly incomplete.

10.10 superformat

```
superformat [-D dos-drive] [-v verbosity-level] [-b begin-track]
[-e end-track] [--superverify] [--dosverify]
[--noverify] [--verify_later] [--zero-based]
[-G format-gap] [-F final-gap] [-i interleave] [-c chunksize]
[-g gap] [--absolute-skew absolute-skew] [--head-skew head-skew]
[--track-skew track-skew] [--biggest-last] drive [media-description]
```

`superformat` is used to format disks with a capacity of up to 1992K HD or 3984K ED. See Chapter 6 [Extended formats], page 13, for a detailed description of these formats. See Chapter 4 [Media description], page 8, for a detailed description of the syntax for the media description. If no media description is given, superformat formats a disk in the highest available density for that drive, using standard parameters (i.e. no extra capacity formats).

When the disk is formatted, `superformat` automatically invokes `mformat` in order to put an MS-DOS filesystem on it. You may ignore this filesystem, if you don't need it.

Supeformat allows to format 2m formats. Be aware, however, that these `2m` formats were specifically designed to hold an MS-DOS filesystem, and that they take advantage of the fact that the MS-DOS filesystem uses redundant sectors on the first track (the FAT, which is represented twice). The second copy of the FAT is *not* represented on the disk.

High capacity formats are sensitive to the exact rotation speed of the drive and the resulting difference in raw capacity. That's why `superformat` performs a measurement of the disks raw capacity before proceeding with the formatting. This measurement is rather time consuming, and can be avoided by storing the relative deviation of the drive capacity into the drive definition file file. See Chapter 5 [Drive descriptions], page 11, for more details on this file. The line to be inserted into the drive definition file is printed by superformat after performing its measurement. However, this line depends on the drive and the controller. Do not copy it to other computers. Remove it before installing another drive or upgrade your floppy controller. Swap the drive numbers if you swap the drives in your computer.

10.10.1 Common Options

Many options have a long and a short form.

`-h`
`--help` Print the help.

`-D` *drive*
`--dosdrive` *dos-drive*
: Selects DOS drive letter for `mformat` (for example `a:` or `b:`). The colon may be omitted. The default is derived from the minor device number. If the drive letter cannot be guessed, and is not given on the command line, `mformat` is skipped.

`-v` *verbosity-level*
`--verbosity` *verbosity-level*
: Sets the verbosity level. 1 prints a dot for each formatted track. 2 prints a changing sign for each formatted track (- for formatting the first head, = for formatting the second head, x for verifying the first head, and + for verifying the second head). 3 prints a complete line listing head and track. 6 and 9 print debugging information.

`--superverify`
: Verifies the disk by first reading the track, than writing a pattern of U's, and then reading it again. This is useful as some errors only show up after the disk has once been written. However, this is also slower.

`-B`
`--dosverify`
: Verifies the disk using the `mbadblocks` program. `mbadblocks` marks the bad sectors as bad in the FAT. The advantage of this is that disks which are only partially bad can still be used for MS-DOS filesystems.

`-V`
`--verify_later`
: Verifies the whole disk at the end of the formatting process instead of at each track. Verifying the disk at each track has the advantage of detecting errors early on.

`-f`
`--noverify`
: Skips the verification altogether.

10.10.2 Advanced Options

Usually, superformat uses sensible default values for these options, which you normally don't need to override. They are intended for expert users. Most of them should only be needed in cases where the hardware or superformat itself has bugs.

`-b` *begin-track*
`--begin_track` *begin-track*
: Describes the track where to begin formatting. This is useful if the previous formatting failed halfway through. The default is 0.

`-e` *end-track*
`--end_track` *end-track*
: Describes where to stop formatting. *end_track* is the last track to be formatted plus one. This is mainly useful for testing purposes. By default, this is the same as the total number of tracks. When the formatting stops, the final skew is displayed (to be used as absolute skew when you'll continue).

`-S` *sizecode*
`--sizecode` *sizecode*
: Set the sector size to be used. The sector size is 128 * (2 ^ *sizecode*). Sector sizes below 512 bytes are not supported, thus sizecode must be at least 2. By

default 512 is assumed, unless you ask for more sectors than would fit with 512 bytes.

`--stretch` *stretch*
: Set the stretch factor. The stretch factor describes how many physical tracks to skip to get to the next logical track (2 ^ *stretch*). On double density 5 1/4 disks, the tracks are further apart from each other.

`-G` *fmt-gap*
`--format_gap` *fmt-gap*
: Set the formatting gap. The formatting gap tells how far the sectors are away from each other. By default, this is chosen so as to evenly distribute the sectors along the track.

`-F` *final-gap*
`--final_gap` *final-gap*
: Set the formatting gap to be used after the last sector.

`-i` *interleave*
`--interleave` *interleave*
: Set the sector interleave factor.

`-c` *chunksize*
`--chunksize` *chunksize*
: Set the size of the chunks. The chunks are small auxiliary sectors used during formatting. They are used to handle heterogeneous sector sizes (i.e. not all sectors have the same size) and negative formatting gaps.

`--biggest-last`
: For MSS formats, make sure that the biggest sector is last on the track. This makes the format more reliable on drives which are out of spec.

`--zero-based`
: Formats the disk with sector numbers starting at 0, rather than 1. Certain CP/M boxes or Music synthesizers use this format. Those disks can currently not be read/written to by the standard Linux read/write API; you have to use fdrawcmd to access them. As disk verifying is done by this API, verifying is automatically switched off when formatting zero-based.

10.10.3 Sector skewing options

In order to maximize the user data transfer rate, the sectors are arranged in such a way that sector 1 of the new track/head comes under the head at the very moment when the drive is ready to read from that track, after having read the previous track. Thus the first sector of the second track is not necessarily near the first sector of the first track. The skew value describes for each track how far sector number 1 is away from the index mark. This skew value changes for each head and track. The amount of this change depends on how fast the disk spins, and on how much time is needed to change the head or the track.

`--absolute_skew` *absolute-skew*
: Set the absolute skew. This skew value is used for the first formatted track. It is expressed in raw bytes.

`--head_skew` *head-skew*
: Set the head skew. This is the skew added for passing from head 0 to head 1. It is expressed in raw bytes.

`--track_skew` *track-skew*
: Set the track skew. This is the skew added for seeking to the next track. It is expressed in raw bytes.

Example: (absolute skew=3, head skew=1, track skew=2)

```
track 0 head 0: 4,5,6,1,2,3 (skew=3)
track 0 head 1: 3,4,5,6,1,2 (skew=4)

track 1 head 0: 1,2,3,4,5,6 (skew=0)
track 1 head 1: 6,1,2,3,4,5 (skew=1)

track 2 head 0: 4,5,6,1,2,3 (skew=3)
track 2 head 1: 3,4,5,6,1,2 (skew=4)
```

N.B. For simplicitie's sake, this example expresses skews in units of sectors. In reality, superformat expects the skews to be expressed in raw bytes.

10.10.4 Examples

In all the examples of this section, we assume that drive 0 is a 3 1/2 and drive 1 a 5 1/4.

The following example shows how to format a 1440K disk in drive 0:

```
superformat /dev/fd0 hd
```

The following example shows how to format a 1200K disk in drive 1:

```
superformat /dev/fd1 hd
```

The following example shows how to format a 1440K disk in drive 1:

```
superformat /dev/fd1 hd sect=18
```

The following example shows how to format a 720K disk in drive 0:

```
superformat /dev/fd0 dd
```

The following example shows how to format a 1743K disk in drive 0 (83 cylinders times 21 sectors):

```
superformat /dev/fd0 sect=21 cyl=83
```

The following example shows how to format a 1992K disk in drive 0 (83 cylinders times 2 heads times 12 KB per track)

```
superformat /dev/fd0 tracksize=12KB cyl=83 mss
```

The following example shows how to format a 1840K disk in drive 0. It will have 5 2048-byte sectors, one 1024-byte sector, and one 512-byte sector per track:

```
superformat /dev/fd0 tracksize=23b mss 2m ssize=2KB
```

All these formats can be autodetected by mtools, using the floppy driver's default settings.

10.10.5 Troubleshooting

`FDC busy, sleeping for a second`
: When another program accesses a disk drive on the same controller as the one being formatted, `superformat` has to wait until the other access is finished. If this happens, check whether any other program accesses a drive (or whether a drive is mounted), kill that program (or unmount the drive), and the format should proceed normally.

`I/O errors during verification`
: Your drive may be too far out of tolerance, and you may thus need to supply a margin parameter. Run `floppymeter` (see Section 10.6 [floppymeter], page 45) to find out an appropriate value for this parameter, and add the suggested `margin` parameter to the command line

10.10.6 Bugs

Opening up new window while `superformat` is running produces overrun errors. These errors are benign, as the failed operation is automatically retried until it succeeds.

10.11 xdfcopy

> `xdfcopy [-`*`format-id`*`] [-d] [-n] [-h `*`head-skew`*`] [-t `*`cylinder-skew`*`] [-T `*`end-cylinder`*`] [`*`source`*`] `*`target`*

`Xdfcopy` is a utility to copy and format XDF disks. XDF (eXtended Density Format) is a format used by OS/2 which can hold 1840KB of data (on a 3 1/2 high density disk). Its advantage over 2m formats is that it is faster: 38KB/s. Because of this fast speed, I extended the XDF standard to higher capacities (1992KB) with a transfer rate of 45KB/s. I called the new formats XXDF.

This program works best with kernels newer than 2.0.0.

If both source and target are given, xdfcopy copies the disk image from file to floppy disk or vice-versa. When copying to a floppy disk, the disk is first formatted, unless the `-n` option is given.

If no source is given, the target is only formatted. In this case, the target must be a floppy drive.

10.11.1 Options

10.11.1.1 Selecting a format

Formats are selected by the format_id. The following formats are understood:

`0`
: Formats a 5 1/4 XDF disk (1520 KB, 45.6 KB/s).

`1`
: Formats a 3 1/2 high density XDF disk (1840 KB, 38.3 KB/s).

`2` Formats a 3 1/2 extra density XDF disk (3680 KB, 102 KB/s)

`3` Formats a 3 1/2 high density XXDF disk (1920 KB, 45 KB/s)

`4` Formats a 3 1/2 extra density XXDF disk (3840 KB, 90 KB/s)

10.11.2 Misc options

`-D` *`dosdrive`*

Describes the DOS drive letter for mformat. If this option is given, an MS-DOS filesystem is automatically installed on the disk after the low-level format is complete. In order for this to work, the drive has to be configured to accept the 23x2x80 geometry in your /etc/mtools or your ~/.mtoolsrc file. Moreover, this only works with a version of mtools that is more recent than 3.0.

Example of a working mtoolsrc line:

```
A /dev/fd0  0  0 0  0
```

Examples of a non-working mtoolsrc line:

```
A /dev/fd0 12 80 2 18
```

`-n` Don't format the disk before copying the disk image to the disk.

10.11.3 Options for power users

`-t` *`cylinder skew`*

Uses a different track skew than the default (14). For more details on skews, see Section 10.10 [superformat], page 48. In this version of xdfcopy, the `-t` parameter is ignored.

`-h` *`head skew`*

Uses a different head skew than the default (0) In this version, this parameter is ignored

`-d` Debugging. For each read or write operation, the time it took to complete the operation is printed (in milliseconds). This can be used to optimize the skews.

`-T` *`end-cylinders`*

Tells how many cylinders to format. With the XXDF formats, it is actually possible to format up to 83 cylinders, yielding a format of up to 1992KB on a 3 1/2 high density disk.

11 Compile-time configuration via GNU autoconf

Before it can be compiled, fdutils must be configured using the GNU autoconf script `./configure`. In most circumstances, running `./configure` without any parameters is enough. However, you may customize fdutils using various options to `./configure`. The following options are supported:

`--prefix` *`directory`*
: Prefix used for any directories used by fdutils. By default, this is '`/usr/local`'. Fdutils is installed in '`$prefix/lib`', looks for its system wide configuration file in '`$prefix/etc`'. Man pages are installed in '`$prefix/man`', info pages in '`$prefix/info`' etc.

`--sysconfdir` *`directory`*
: Directory containing the system-wide configuration files such as '`mediaprm`' and '`driveprm`'. By default, this is derived from `prefix` (see above). N.B. For backward compatibility reasons, the old-style floppy parameter file '`fdprm`' is always in '`/etc`', regardless of the setting for `sysconfdir`

`--enable-fdmount-floppy-only`
: Allows usage of the `fdmount` program only to members of the group `floppy`

In addition to the above-listed options, the other standard GNU autoconf options apply. Type `./configure --help` to get a complete list of these.

Appendix A Acronyms

SD (Single density)
: Single density disks use a data transfer rate of 125 kb/s and are no longer in use today, because of their low capacity.

DD (Double density)
: Double density disks normally hold 360KB (5 1/4) or 720KB (3 1/2) of data per disk.

  Double density disks have only a hole on one side (for write protection).

  Double density uses a data transfer rate of 250 kb/s or 300 kb/s depending on the drive type: 5 1/4 high density drives use 300 kb/s when writing to a double density diskm 250 kb/s is used in all other circumstances. The reason why double density disks in high density drives need a higher data transfer rate is because these drives rotate faster (360 rpm instead of 300 rpm).

HD (High density)
: High density disks normally hold 1200KB (5 1/4) or 1440KB (3 1/2) of data per disk. High density 3 1/2 disks are marked as such by the presence of a second square hole, just opposed to the write protect hole. 3 1/2 high density disks are the most commonly used type of disks today.

QD (Quad density)
: Quad density is a hybrid between double and high density. It only exists for 5 1/4 disks, and holds 720KB of data. It can be obtained by formatting DD disks in a HD drive. QD uses double density for the density along the tracks (data transfer rate), and high density for the density perpendicular to the tracks (spacing between tracks, and thus number of tracks). This came to existence because these two aspects are limited by two different factors: the density along the track is limited by the quality of the media, whereas the density perpendicular to the tracks is mainly limited by the drive mechanism (this

density, expressed in bits per inch comes nowhere near the limits of the media, even with HD). Thus quad density is an easy way to double the capacity of an ordinary double density disk, just by formatting it in a HD drive.

ED (Extra density)
: Extra density refers to a disk density that can normally hold 2880KB of data per disk. Extra density disks only exist as 3 1/2 disks. ED disks are marked with a second squared hole opposed to the square hole, which is a little bit closer to the middle of the edge than that of HD disks. This format never really took off, because it only was released when storage media with a much higher capacity, such as CD-Roms, tapes and Zip disks became popular.

  ED uses a data transfer rate of 250 kb/s.

DS (Double sided)
: Self explanatory.

SS (Single sided)
: Self explanatory

MSS (Mixed size sectors)
: Mixed sector size formats are formats which use sectors of several different sizes on a single track. See Section 6.6 [Mixed size sectors], page 17, for details.

2M (2 Megabytes)
: 2M is a high capacity format developped by Ciriaco de Celis. The basic principle is the same as MSS: mix sectors of several sizes on a same track, in order to minimize both slack space and header overhead. 2M is different from MSS in that it uses a normal 18 sector format on its first track. See Section 6.8 [2M], page 18, for details.

rpm (Rotations per minute)
: All 3 1/2 drives and 5 1/4 DD drives run at 300 rotations per minute, whereas 5 1/4 HD drives run at 360 rotations per minute.

rps (Rotations per second)
: See above.

tpi (tracks per inch)
: Expresses how close cylinders are to each other. Usually, 5 1/4 double density disks have 48 tpi, whereas 5 1/4 high density and quad density disks have 96 tpi. 3 1/2 disks use 135.5 tpi.

XDF (eXtended Density Format)
: XDF is a disk format used for the OS/2 distribution disks. Its operating systems are similar to 2M and MSS disk, but it is faster due to a more creative arrangement of sectors. See Section 6.9 [XDF], page 19, for details.

XXDF (eXtended XDF)
: XXDF is an Linux enhancement for XDF. It can store 1992 KB of data on an ED disk instead of just 1840 available with the regular XDF format. See Section 6.10 [XXDF], page 20, for details.

MFM (Multi Frequency Modulation)
: MFM is a low level encoding of disk data. It is used for DD, HD and ED disks, i.e. virtually all disks that are available today. The PC hardware can only read MFM and FM disks. The doc at:

  `http://www.moria.de/~michael/floppy/floppy.ps`

  contains more detailed information about FM and MFM encoding.

FM (Frequency modulation)
: FM is a low level encoding of disk data. It was used for SD disks, and is now considered to be obsolete. The doc at:

  `http://www.moria.de/~michael/floppy/floppy.ps`

  contains more detailed information about FM and MFM encoding.

kb (kilobit)
: 1000 bits

kb/s (kilobit per second)
: We express the raw data throughput to and from the disk in this unit, which is also used in the documentation of the floppy disk controller.

B
: Byte. A byte is 8 bits, and is the smallest individually addressable unit of data.

KB (K-Byte)
: 1024 bytes. Sometimes also noted K.

KB/s (K-Byte)
: We express the usable data throughput to and from the disk in KB/s. Roughly, 1 KB/s = 8 kb/s. However, the usable data throughput is always lower than the raw throughput due to header overhead, interleaving and seek overhead.

MB (Megabyte)
: Initially, 1 megabyte was 1024*1024 bytes (i.e. 1048576 bytes). However, when talking of floppy disk capacity, we understand it as 1000KB, that is 1000*1024 bytes, i.e. 1024000 bytes.

MB/s (million bytes per second)
: We express (high) raw data throughput to and from the disk in kb/s, which is also used in the documentation of the floppy disk controller.

Appendix B Interesting formats

This table lists a few interesting formats. Some of them are not included in the table of predefined formats due to lack of space. You may use these by setting the geometry of the variable geometry device using `setfdprm`. Most of the entries in the following table do not describe a single format, but rather a family of formats which share a common characteristic. For this reason, no precise parameters are included.

80/90/160/180 KB 5.25"
: Original IBM PC, CP/M, Apple II, TRS-80, etc. Single density, FM (not MFM), 40-track. Some systems also used "hard" sectoring, with an index hole for each sector. These are probably readable on standard FDCs if anyone cares to go to the effort. 160KB/180KB formats were double-sided.

320 KB 5.25"

CP/M, Zenith Z-100, some forms of MS-DOS. 8-sector DD 40-track.

400 KB 5.25"

AT&T 7300/3B1: cpio format, 10 sectors, DS, 40 tracks.

PC: MS-DOS, 10 sectors, DS, 40 tracks; formatted by various utilities; readable by some versions of MS-DOS.

<720KB 3.5" formats

- Some dedicated word-processing systems use lower than normal capacity floppies (some may be 40-track either via "stretching" or by having odd drives; others may be single-sided or use fewer sectors per track, or use "single density" or FM).
- Atari ST 360 KB floppies: these are 9 sector 250kb/s 80-track single-sided, using a slightly incompatible MS-DOS-like FAT. Very common distribution format for Atari ST software.
- Mac 400 KB: single-sided 80-tracks, Mac MFS or HFS file system, Group Code Recording, zone bit recording, variable RPMs from 300 to 600 in 5 zones, and auto-eject mechanism. Impossible to read on normal PC drives without some form of hardware assistance.

720 KB 3.5"

- PC: 9 sectors, DS, 80-track; very common.
- Atari ST: same as Atari ST 360 KB except double-sided.

720 KB 5.25"

"Quad Density": 9 sectors, DS, 80-track; name is rather misleading, as it is double density 80 track.

800 KB

- PC: 10 sectors, DS, 80 tracks; readable by MS-DOS; formatted by many PC utilities; "almost standard".
- Atari ST "Twister": 10 sectors, DS, 80-track; slightly incompatible MS-DOS-like FAT; sector skewing (originated by David Small).
- Mac 800 KB: same as Mac 400KB except double-sided.

880 KB

- PC: 11 sectors, DS, 80 tracks, interleaved(?). Formatted by various PD utilities such as fdformat.
- Amiga: uses "single-sector" tracks the size of 11 sectors; DS; 80 tracks. Written by a custom chip rather than a standard FDC; it is very difficult to access these using a standard FDC, though it might be possible in theory, using some trickery.

880 KB - 960 KB

Atari ST: 11/12 sectors, DS, 80-track, interleaved(?); not very reliable, formatted by various PD utilities.

960 KB 2m and Linux; 6 sectors interleaved(?), DS, 250kb/s?, 80-track, 1K sectors(?).

1001 KB "Japanese" format: 300kb/s, 77 tracks, HD drives/disks.

1040 KB 2m and Linux only; 13 sectors, DS, 300kb/s, 80-track.

1120 KB 2m and Linux only; 14 sectors interleaved, DS, 300kb/s, 80-track. Can only be formatted on Linux; even on Linux, it is difficult to format these on an ED drive.

1200 KB 5.25" HD
"AT": 360 rpm 500kb/s 80-track 15-sector MS-DOS.

1360 KB 5.25" HD
Highest "fdformat" noninterleaved 5.25" format. 360 rpm 500kb/s 17-sector MS-DOS.

1440 KB 5.25" HD
Interleaved 18-sector 360 rpm 500kb/s; highest "fdformat" 5.25" format. 18-sector MS-DOS.

2m and Linux only: noninterleaved 1024-byte sector 18-sector-equivalent 360 rpm 500kb/s; 18-sector MS-DOS. A good substitute for 1.44MB 3.5" floppies.

1600 KB 5.25" HD
2m and Linux only: interleaved? 500kb/s 80-track; 1 8KB sector, 1 2KB sector?

1760 KB 3.5" HD
2m and Linux only: 11 1KB sectors; 500kb/s, noninterleaved?

1840 KB 3.5" HD
2m and Linux only: 23-sector equivalent; 500kb/s, noninterleaved.

1920 KB 3.5" HD
2m and Linux only: 3 4KB sectors; 500kb/s, interleaved?

2m and Linux only: 1 8KB sector, 1 4KB sector; 500kb/s, interleaved?

2880 KB 3.5" ED
"Extra Density" or "Extra High Density". 1 Mb/s, 36-sector, DS, 80- track.

3200 KB 3.5" ED
Non-interleaved, 80 track, 40-sector 1 Mb/s. Highest capacity that MS-DOS can read directly(?)

3520 KB 3.5" ED
2m and Linux only: 1 16KB sector, 1 4KB sector, 1 2KB sector? Non-interleaved, 80 track 1 Mb/s. Highest capacity that 2m can format.

3840 KB 3.5" ED
2m and Linux only: 1 16KB sector, 1 8KB sector (or is it 3 8KB sectors?). Non-interleaved, 80 track 1 Mb/s. Formatted only by Linux, but readable and writeable by 2m.

Linux is able to read almost any MFM disk. These include many CP/M disks and also Commodore 1581 disks. Please get Michael Haardt's documentation on floppy drives for a detailed description of those formats. This can be ftp'ed from the following location:

```
http://www.moria.de/~michael/floppy/
```

Commodore 1581 disks are not yet described in this documentation. Use `setfdprm /dev/fd0 DD DS sect=10 cyl=80 swapsides`. If you want to use these disks often, redefine one of the "default" formats to be Commodore 1581, and then put it into the autodetection list for the drive. The following example describes how to redefine format number 31 (minor device number 124) to be Commodore 1581:

```
mknod /dev/fd0cbm1581 b 2 124
setfdprm /dev/fd0cbm1581 DD DS sect=10 cyl=80 swapsides
floppycontrol --autodetect /dev/fd0 31,7,8,4,25,28,22,21
```

The two latter commands have to be issued after each reboot, so I suggest you put them into your '`/etc/rc`' files if you use many Commodore 1581 disks.

Command Index

Concept index

T

V

X

Table of Contents