NAME

mkfs.fat − create an MS-DOS FAT filesystem

SYNOPSIS

mkfs.fat [*OPTIONS*] *DEVICE* [*BLOCK-COUNT*]

DESCRIPTION

mkfs.fat is used to create a FAT filesystem on a device or in an image file. *DEVICE* is the special file corresponding to the device (e.g. /dev/sdXX) or the image file (which does not need to exist when the option **-C** is given). *BLOCK-COUNT* is the number of blocks on the device and size of one block is always 1024 bytes, independently of the sector size or the cluster size. Therefore *BLOCK-COUNT* specifies size of filesystem in KiB unit and not in the number of sectors (like for all other **mkfs.fat** options). If omitted, **mkfs.fat** automatically chooses a filesystem size to fill the available space.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

In Atari mode, if not directed otherwise by the user, **mkfs.fat** will always use 2 sectors per cluster, since GEMDOS doesn't like other values very much. It will also obey the maximum number of sectors GEMDOS can handle. Larger filesystems are managed by raising the logical sector size. An Atari-compatible serial number for the filesystem is generated, and a 12 bit FAT is used only for filesystems that have one of the usual floppy sizes (720k, 1.2M, 1.44M, 2.88M), a 16 bit FAT otherwise. This can be overridden with the **−F** option. Some PC-specific boot sector fields aren't written, and a boot message (option **−m**) is ignored.

OPTIONS

−a Normally, for any filesystem except very small ones, **mkfs.fat** will align all the data structures to cluster size, to make sure that as long as the partition is properly aligned, so will all the data structures in the filesystem. This option disables alignment; this may provide a handful of additional clusters of storage at the expense of a significant performance degradation on RAIDs, flash media or large-sector hard disks.

−A Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.

−b *SECTOR-OF-BACKUP*
Selects the location of the backup boot sector for FAT32. Default depends on number of reserved sectors, but usually is sector 6. If there is a free space available after the backup boot sector then backup of the FAT32 info sector is put after the backup boot sector, usually at sector 7. The backup must be within the range of reserved sectors. Value 0 completely disables creating of backup boot and info FAT32 sectors.

−c Check the device for bad blocks before creating the filesystem.

−C Create the file given as *DEVICE* on the command line, and write the to-be-created filesystem to it. This can be used to create the new filesystem in a file instead of on a real device, and to avoid using **dd** in advance to create a file of appropriate size. With this option, the *BLOCK-COUNT* must be given, because otherwise the intended size of the filesystem wouldn't be known. The file created is a sparse file, which actually only contains the meta-data areas (boot sector, FATs, and root directory). The data portions won't be stored on the disk, but the file nevertheless will have the correct size. The resulting file can be copied later to a floppy disk or other device, or mounted through a loop device.

−D *DRIVE-NUMBER*
Specify the BIOS drive number to be stored in the FAT boot sector. For hard disks and removable medias it is usually 0x80–0xFF (0x80 is first hard disk C:, 0x81 is second hard disk D:, ...), for floppy devices or partitions to be used for floppy emulation it is 0x00–0x7F (0x00 is first floppy A:, 0x01 is second floppy B:).

−f *NUMBER-OF-FATS*
Specify the number of file allocation tables in the filesystem. The default is 2.

−F *FAT-SIZE*
Specifies the type of file allocation tables used (12, 16 or 32 bit). If nothing is specified, **mkfs.fat** will automatically select between 12, 16 and 32 bit, whatever fits better for the filesystem size.

−g *HEADS*/*SECTORS-PER-TRACK*
Specify *HEADS* and *SECTORS-PER-TRACK* numbers which represents disk geometry of *DEVICE*. Both numbers are stored into the FAT boot sector. Number*SECT ORS-PER-TRACK* is used also for aligning the total count of FAT sectors. By default disk geometry is read from *DEVICE* itself. If it is not available then *LBA-Assist Translation* and translation table from the *SD Card Part 2 File System Specification* based on total number of disk sectors is used.

−h *NUMBER-OF-HIDDEN-SECTORS*
Specify the number of so-called *hidden sectors*, as stored in the FAT boot sector: this number represents the beginning sector of the partition containing the file system. Normally this is an offset (in sectors) relative to the start of the disk, although for MBR logical volumes contained in an extended partition of type 0x05 (a non-LBA extended partition), a quirk in the MS-DOS implementation of FAT requires it to be relative to the partition’s immediate containing Extended Boot Record. Boot code and other software handling FAT volumes may also rely on this field being set up correctly; most modern FAT implementations will ignore it. By default, if the *DEVICE* is a partition block device, **mkfs.fat** uses the partition offset relative to disk start. Otherwise, **mkfs.fat** assumes zero. Use this option to override this behaviour.

−i *VOLUME-ID*
Sets the volume ID of the newly created filesystem; *VOLUME-ID* is a 32-bit hexadecimal number (for example, 2e24ec82). The default is a number which depends on the filesystem creation time.

−I Ignore and disable safety checks. By default **mkfs.fat** refuses to create a filesystem on a device with partitions or virtual mapping. **mkfs.fat** will complain and tell you that it refuses to w ork. This is different when using MO disks. One doesn’t always need partitions on MO disks. The filesystem can go directly to the whole disk. Under other OSes this is known as the *superfloppy* format. This switch will force **mkfs.fat** to work properly.

−l *FILENAME*
Read the bad blocks list from *FILENAME*.

−m *MESSAGE-FILE*
Sets the message the user receives on attempts to boot this filesystem without having properly installed an operating system. The message file must not exceed 418 bytes once line feeds have been converted to carriage return-line feed combinations, and tabs have been expanded. If the filename is a hyphen (-), the text is taken from standard input.

−M *FAT-MEDIA-TYPE*
Specify the media type to be stored in the FAT boot sector. This value is usually 0xF8 for hard disks and is 0xF0 or a value from 0xF9 to 0xFF for floppies or partitions to be used for floppy emulation.

−−mbr[=*y*|*yes*|*n*|*no*|*a*|*auto*]
Fill (fake) MBR table with disk signature one partition which starts at sector 0 (includes MBR itself) and spans whole disk device. It is needed only for non-removable disks used on Microsoft Windows systems and only when formatting whole unpartitioned disk. Location of the disk signature and partition table overlaps with the end of the first FAT sector (boot code location), therefore there is no additional space usage. Default is *auto* mode in which **mkfs.fat** put MBR table only for non-removable disks when formatting whole unpartitioned disk.

−n *VOLUME-NAME*
Sets the volume name (label) of the filesystem. The volume name can be up to 11 characters long. Supplying an empty string, a string consisting only of white space or the string "NO NAME" as *VOLUME-NAME* has the same effect as not giving the **−n** option. The default is no label.

−−codepage=*PAGE*
Use DOS codepage *PAGE* to encode label. By default codepage 850 is used.

−r *ROOT-DIR-ENTRIES*
Select the minimal number of entries available in the root directory. The default is 112 or 224 for floppies and 512 for hard disks. Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **−a**.

−R *NUMBER-OF-RESERVED-SECTORS*
Select the minimal number of reserved sectors. With FAT32 format at least 2 reserved sectors are needed, the default is 32. Otherwise the default is 1 (only the boot sector). Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **−a**.

−s *SECTORS-PER-CLUSTER*
Specify the number of disk sectors per cluster. Must be a power of 2, i.e. 1, 2, 4, 8, ... 128.

−S *LOGICAL-SECTOR-SIZE*
Specify the number of bytes per logical sector. Must be a power of 2 and greater than or equal to 512, i.e. 512, 1024, 2048, 4096, 8192, 16384, or 32768. Values larger than 4096 are not conforming to the FAT file system specification and may not work everywhere.

−v Verbose execution.

−−offset *SECTOR*
Write the filesystem at a specific sector into the device file. This is useful for creating a filesystem in a partitioned disk image without having to set up a loop device.

−−variant *TYPE*
Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

−−help
Display option summary and exit.

−−invariant
Use constants for normally randomly generated or time based data such as volume ID and creation time. Multiple runs of**mkfs.fat** on the same de vice create identical results with this option. Its main purpose is testing **mkfs.fat**.

BUGS

mkfs.fat can not create boot-able filesystems. This isn't as easy as you might think at first glance for various reasons and has been discussed a lot already. **mkfs.fat** simply will not support it ;)

SEE ALSO

fatlabel(8), **fsck.fat**(8)

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page ⟨https://github.com/dosfstools/dosfstools⟩.

AUTHORS

dosfstools were written by Werner Almesberger ⟨werner.almesberger@lrc.di.epfl.ch⟩, Roman Hodek ⟨Roman.Hodek@informatik.uni-erlangen.de⟩, and others. Current maintainers are Andreas Bombe ⟨aeb@debian.org⟩ and Pali Rohár ⟨pali.rohar@gmail.com⟩.