NAME

mkfs.fat - create an MS-DOS filesystem under Linux

SYNOPSIS

mkfs.fat [**-a**] [**-A**] [**-b** *sector-of-backup*] [**-c**] [**-l** *filename*] [**-C**] [**-f** *number-of-FATs*] [**-F** *FAT-size*] [**-h** *number-of-hidden-sectors*] [**-i** *volume-id*] [**-I**] [**-m** *message-file*] [**-n** *volume-name*] [**-r** *root-dir-entries*] [**-R** *number-of-reserved-sectors*] [**-s** *sectors-per-cluster*] [**-S** *logical-sector-size*] [**-D** *drive-number*] [**-M** *FAT-media-type*] [**-v**] *device* [*block-count*]

DESCRIPTION

mkfs.fat is used to create an MS-DOS filesystem under Linux on a device (usually a disk partition). *device* is the special file corresponding to the device (e.g /dev/sdXX). *block-count* is the number of blocks on the device. If omitted, **mkfs.fat** automatically determines the filesystem size.

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

Use Atari variation of the MS-DOS filesystem. This is default if **mkfs.fat** is run on an Atari, then this option turns off Atari format. There are some differences when using Atari format: 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. Under Atari format, 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.

-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. The backup must be within the range of reserved 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. This value is usually 0x80 for hard disks and 0x00 for floppy devices or partitions to be used for floppy emulation.

-f *number-of-FATs*

Specify the number of file allocation tables in the filesystem. The default is 2. Currently the Linux MS-DOS filesystem does not support more than 2 FATs.

-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.

-h *number-of-hidden-sectors*
Select the number of hidden sectors in the volume. Apparently some digital cameras get indigestion if you feed them a CF card without such hidden sectors, this option allows you to satisfy them.

-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 It is typical for fixed disk devices to be partitioned so, by default, you are not permitted to create a filesystem across the entire device. **mkfs.fat** will complain and tell you that it refuses to work. 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 has a value from 0xF9 to 0xFF for floppies or partitions to be used for floppy emulation.

-n *volume-name*
Sets the volume name (label) of the filesystem. The volume name can be up to 11 characters long. The default is no label.

-r *root-dir-entries*
Select the number of entries available in the root directory. The default is 112 or 224 for floppies and 512 for hard disks.

-R *number-of-reserved-sectors*
Select the 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).

-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.

-v Verbose execution.

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

More information about **fsck.fat** and **dosfstools** can be found at <*http://daniel-baumann.ch/software/dosfstools/*>.

AUTHORS

dosfstools were written by Werner Almesberger <*werner.almesberger@lrc.di.epfl.ch*>, Roman Hodek <*Roman.Hodek@informatik.uni-erlangen.de*>, and others. The current maintainer is Daniel Baumann <*mail@daniel-baumann.ch*>.