NAME

newfs_msdos - construct a new MS-DOS (FAT) file system

SYNOPSIS

newfs_msdos [**-N**] [**-@** *offset*] [**-A**] [**-B** *boot*] [**-C** *create-size*] [**-F** *FAT-type*] [**-I** *VolumeID*] [**-L** *label*] [**-O** *OEM*] [**-S** *sector-size*] [**-T** *timestamp*] [**-a** *FAT-size*] [**-b** *block-size*] [**-c** *cluster-size*] [**-e** *DirEnts*] [**-f** *format*] [**-h** *heads*] [**-i** *info*] [**-k** *backup*] [**-m** *media*] [**-n** *FATs*] [**-o** *hidden*] [**-r** *reserved*] [**-s** *total*] [**-u** *track-size*] *special* [*disktype*]

DESCRIPTION

The **newfs_msdos** utility creates a FAT12, FAT16, or FAT32 file system on device or file named *special*, using disktab(5) entry *disktype* to determine geometry, if required.

If *special* does not contain a */* and **-C** is not used, it is assumed to be a device name and */dev/* is prepended to the name to construct the actual device name. To work a file in the current directory use *./filename*

The options are as follow:

-N Do not create a file system: just print out parameters.

-@ *offset*

Build the filesystem at the specified offset in bytes in the device or file. A suffix s, k, m, g (lower or upper case) appended to the offset specifies that the number is in sectors, kilobytes, megabytes or gigabytes, respectively.

-A Attempt to cluster align root directory, useful for SD card.

-B *boot*

Get bootstrap from file.

-C *create-size*

Create the image file with the specified size. A suffix character appended to the size is interpreted as for the **-@** option. The file is created by truncating any existing file with the same name and resizing it to the requested size. If the file system supports sparse files, the space occupied on disk may be smaller than the size specified as parameter.

-F *FAT-type*

FAT type (one of 12, 16, or 32).

-I *VolumeID*

Volume ID, a 32 bit number in decimal or hexadecimal (0x...) format.

-L *label*

Volume label (up to 11 characters). The label should consist of only those characters permitted in regular DOS (8+3) filenames.

-O *OEM*

OEM string (up to 8 characters). The default is "BSD4.4 ".

-S *sector-size*

Number of bytes per sector. Acceptable values are powers of 2 in the range 512 through 32768, inclusive.

-T *timestamp*

Create the filesystem as though the current time is *timestamp*. The default filesystem volume ID is derived from the time. *timestamp* can be a pathname (where the timestamp is derived from that file) or an integer value interpreted as the number of seconds since the Epoch.

-a *FAT-size*

Number of sectors per FAT.

-b *block-size*

File system block size (bytes per cluster). This should resolve to an acceptable number of sectors per cluster (see below).

-c *cluster-size*

Sectors per cluster, also called allocation size. Acceptable values are powers of 2 in the range 1 through 128. If the block or cluster size are not specified, the code uses a cluster between 512 bytes and 32K depending on the filesystem size.

-e *DirEnts*

Number of root directory entries (FAT12 and FAT16 only).

-f *format*

Specify a standard (floppy disk) format. The standard formats are (capacities in kilobytes): 160, 180, 320, 360, 640, 720, 1200, 1232, 1440, 2880.

-h *heads*

Number of drive heads.

-i *info* Location of the file system info sector (FAT32 only). A value of 0xffff signifies no info sector.

-k *backup*
Location of the backup boot sector (FAT32 only). A value of 0xffff signifies no backup sector.

-m *media*
Media descriptor (acceptable range 0xf0 to 0xff).

-n *FATs*
Number of FATs. Acceptable values are 1 to 16 inclusive. The default is 2.

-o *hidden*
Number of hidden sectors.

-r *reserved*
Number of reserved sectors.

-s *total*
File system size.

-u *track-size*
Number of sectors per track.

NOTES

If some parameters (e.g., size, number of sectors, etc.) are not specified through options or disktype, the program tries to generate them automatically. In particular, the size is determined as the device or file size minus the offset specified with the **-@** option. When the geometry is not available, it is assumed to be 63 sectors, 255 heads. The size is then rounded to become a multiple of the track size and avoid complaints by some filesystem code.

FAT file system parameters occupy a "Boot Sector BPB (BIOS Parameter Block)" in the first of the "reserved" sectors which precede the actual file system. For reference purposes, this structure is presented below.

```
struct bsbpb {
    uint16_t        bpbBytesPerSec;         /* [-S] bytes per sector */
    uint8_t bpbSecPerClust;         /* [-c] sectors per cluster */
    uint16_t        bpbResSectors;          /* [-r] reserved sectors */
    uint8_t bpbFATs;                /* [-n] number of FATs */
    uint16_t        bpbRootDirEnts;         /* [-e] root directory entries */
```

```
    uint16_t    bpbSectors;         /* [-s] total sectors */
    uint8_t bpbMedia;           /* [-m] media descriptor */
    uint16_t    bpbFATsecs;         /* [-a] sectors per FAT */
    uint16_t    bpbSecPerTrack;     /* [-u] sectors per track */
    uint16_t    bpbHeads;           /* [-h] drive heads */
    uint32_t    bpbHiddenSecs;      /* [-o] hidden sectors */
    uint32_t    bpbHugeSectors;     /* [-s] big total sectors */
};
/* FAT32 extensions */
struct bsxbpb {
    uint32_t    bpbBigFATsecs;      /* [-a] big sectors per FAT */
    uint16_t    bpbExtFlags;        /* control flags */
    uint16_t    bpbFSVers;          /* file system version */
    uint32_t    bpbRootClust;       /* root directory start cluster */
    uint16_t    bpbFSInfo;          /* [-i] file system info sector */
    uint16_t    bpbBackup;          /* [-k] backup boot sector */
};
```

LIMITATION

The maximum file size is 4GB, even if the file system itself is bigger.

EXIT STATUS

Exit status is 0 on success and 1 on error.

EXAMPLES

Create a file system, using default parameters, on */dev/ada0s1*:

```
newfs_msdos /dev/ada0s1
```

Create a FAT32 filesystem with a 32K allocation size on */dev/mmcsd0s1*:

```
newfs_msdos -F 32 -A -c 64 /dev/mmcsd0s1
```

Create a standard 1.44M file system, with volume label *foo*, on */dev/fd0*:

```
newfs_msdos -f 1440 -L foo fd0
```

Create a 30MB image file, with the FAT partition starting 63 sectors within the image file:

```
newfs_msdos -C 30M -@63s ./somefile
```

SEE ALSO

gpart(8), newfs(8)

HISTORY

The **newfs_msdos** utility first appeared in FreeBSD 3.0.

AUTHORS

Robert Nordier <*rnordier@FreeBSD.org*>