Next Previous Contents

1. FAT

The traditional DOS filesystem types are FAT12 and FAT16. Here FAT stands for File Allocation Table: the disk is divided into *clusters*, the unit used by the file allocation, and the FAT describes which clusters are used by which files.

Let us describe the FAT filesystem in some detail. The FAT12/16 type is important, not only because of the traditional use, but also because it is useful for data exchange between different operating systems, and because it is the filesystem type used by all kinds of devices, like digital cameras.

1.1 Layout

First the boot sector (at relative address 0), and possibly other stuff. Together these are the Reserved Sectors. Usually the boot sector is the only reserved sector.

Then the FATs (following the reserved sectors; the number of reserved sectors is given in the boot sector, bytes 14-15; the length of a sector is found in the boot sector, bytes 11-12).

Then the Root Directory (following the FATs; the number of FATs is given in the boot sector, byte 16; each FAT has a number of sectors given in the boot sector, bytes 22-23).

Finally the Data Area (following the root directory; the number of root directory entries is given in the boot sector, bytes 17-18, and each directory entry takes 32 bytes; space is rounded up to entire sectors).

1.2 Boot sector

The first sector (512 bytes) of a FAT filesystem is the *boot sector*. In Unix-like terminology this would be called the superblock. It contains some general information.

First an explicit example (of the boot sector of a DRDOS boot floppy).

```
0000000 eb 3f 90 49 42 4d 20 20 33 2e 33 00 02 01 01 00
0000020 02 e0 00 40 0b f0 09 00 12 00 02 00 00 00 00 00
0000040 00 00 00 00 00 00 00 00 00 00 70 00 ff ff 49 42
0000060 4d 42 49 4f 20 20 43 4f 4d 00 50 00 00 08 00 18
...
```

(See here for the complete sector. And also a MSDOS example)

The 2-byte numbers are stored little endian (low order byte first).

Here the FAT12 version, that is also the common part of the FAT12, FAT16 and FAT32 boot sectors. See further below.

```
Bytes   Content
```

```
 0-2     Jump to bootstrap (E.g. eb 3c 90; on i86: JMP 003E NOP.
         One finds either eb xx 90, or e9 xx xx.
         The position of the bootstrap varies.)
 3-10    OEM name/version (E.g. "IBM  3.3", "IBM 20.0", "MSDOS5.0", "MSWIN4.0".
         Various format utilities leave their own name, like "CH-FOR18".
         Sometimes just garbage. Microsoft recommends "MSWIN4.1".)
         /* BIOS Parameter Block starts here */
 11-12   Number of bytes per sector (512)
         Must be one of 512, 1024, 2048, 4096.
 13      Number of sectors per cluster (1)
         Must be one of 1, 2, 4, 8, 16, 32, 64, 128.
         A cluster should have at most 32768 bytes. In rare cases 65536 is OK.
 14-15   Number of reserved sectors (1)
         FAT12 and FAT16 use 1. FAT32 uses 32.
 16      Number of FAT copies (2)
 17-18   Number of root directory entries (224)
         0 for FAT32. 512 is recommended for FAT16.
 19-20   Total number of sectors in the filesystem (2880)
         (in case the partition is not FAT32 and smaller than 32 MB)
 21      Media descriptor type (f0: 1.4 MB floppy, f8: hard disk; see below)
 22-23   Number of sectors per FAT (9)
         0 for FAT32.
 24-25   Number of sectors per track (12)
 26-27   Number of heads (2, for a double-sided diskette)
 28-29   Number of hidden sectors (0)
         Hidden sectors are sectors preceding the partition.
         /* BIOS Parameter Block ends here */
 30-509  Bootstrap
 510-511 Signature 55 aa
```

The signature is found at offset 510-511. This will be the end of the sector only in case the sector size is 512.

Media descriptor byte

The ancient media descriptor type codes are:

```
 For 8" floppies:
 fc, fd, fe - Various interesting formats

 For 5.25" floppies:
 Value  DOS version  Capacity  sides  tracks  sectors/track
 ff     1.1           320 KB    2      40      8
 fe     1.0           160 KB    1      40      8
 fd     2.0           360 KB    2      40      9
 fc     2.0           180 KB    1      40      9
 fb                   640 KB    2      80      8
 fa                   320 KB    1      80      8
 f9     3.0          1200 KB    2      80     15

 For 3.5" floppies:
 Value  DOS version  Capacity  sides  tracks  sectors/track
 fb                   640 KB    2      80      8
 fa                   320 KB    1      80      8
 f9     3.2           720 KB    2      80      9
 f0     3.3          1440 KB    2      80     18
 f0                  2880 KB    2      80     36

 For RAMdisks:
 fa
```

```
For hard disks:
Value  DOS version
f8     2.0
```

This code is also found in the first byte of the FAT.

IBM defined the media descriptor byte as 11111red, where r is removable, e is eight sectors/track, d is double sided.

Cluster size

The default number of sectors per cluster for floppies (with FAT12) is

```
Drive size Secs/cluster   Cluster size
   360 KB       2             1 KiB
   720 KB       2             1 KiB
   1.2 MB       1           512 bytes
  1.44 MB       1           512 bytes
  2.88 MB       2             1 KiB
```

The default number of sectors per cluster for fixed disks is (with FAT12 below 16 MB, FAT16 above):

```
Drive size Secs/cluster   Cluster size
 <  16 MB       8             4 KiB
 < 128 MB       4             2 KiB
 < 256 MB       8             4 KiB
 < 512 MB      16             8 KiB
 <   1 GB      32            16 KiB
 <   2 GB      64            32 KiB
 <   4 GB     128            64 KiB   (Windows NT only)
 <   8 GB     256           128 KiB   (Windows NT 4.0 only)
 <  16 GB     512           256 KiB   (Windows NT 4.0 only)
```

Usually people have file systems with average file size a few KB. With a cluster size of 64 KiB or more the amount of slack (wasted space) can easily become more than half the total space. FAT32 allows larger file systems with a small cluster size:

```
Drive size Secs/cluster   Cluster size
 < 260 MB       1           512 bytes
 <   8 GB       8             4 KiB
 <  16 GB      16             8 KiB
 <  32 GB      32            16 KiB
 <   2 TB      64            32 KiB
```

FAT16

FAT16 uses the above BIOS Parameter Block, with some extensions:

```
11-27   (as before)
28-31   Number of hidden sectors (0)
32-35   Total number of sectors in the filesystem
        (in case the total was not given in bytes 19-20)
36      Logical Drive Number (for use with INT 13, e.g. 0 or 0x80)
37      Reserved (Earlier: Current Head, the track containing the Boot Record)
        Used by Windows NT: bit 0: need disk check; bit 1: need surface scan
```

```
38      Extended signature (0x29)
        Indicates that the three following fields are present.
        Windows NT recognizes either 0x28 or 0x29.
39-42   Serial number of partition
43-53   Volume label or "NO NAME    "
54-61   Filesystem type (E.g. "FAT12   ", "FAT16   ", "FAT     ", or all zero.)
62-509  Bootstrap
510-511 Signature 55 aa
```

FAT32

FAT32 uses an extended BIOS Parameter Block:

```
11-27   (as before)
28-31   Number of hidden sectors (0)
32-35   Total number of sectors in the filesystem
36-39   Sectors per FAT
40-41   Mirror flags
        Bits 0-3: number of active FAT (if bit 7 is 1)
        Bits 4-6: reserved
        Bit 7: one: single active FAT; zero: all FATs are updated at runtime
        Bits 8-15: reserved
42-43   Filesystem version
44-47   First cluster of root directory (usually 2)
48-49   Filesystem information sector number in FAT32 reserved area (usually 1)
50-51   Backup boot sector location or 0 or 0xffff if none (usually 6)
52-63   Reserved
64      Logical Drive Number (for use with INT 13, e.g. 0 or 0x80)
65      Reserved - used to be Current Head (used by Windows NT)
66      Extended signature (0x29)
        Indicates that the three following fields are present.
67-70   Serial number of partition
71-81   Volume label
82-89   Filesystem type ("FAT32   ")
```

The old 2-byte fields "total number of sectors" and "number of sectors per FAT" are now zero; this information is now found in the new 4-byte fields.

An important improvement is the "First cluster of root directory" field. Earlier, the root directory was not part of the Data Area, and was in a known place with a known size, and hence was unable to grow. Now the root directory is just somewhere in the Data Area.

1.3 The File Allocation Table

The disk is divided into clusters. The number of sectors per cluster is given in the boot sector byte 13.

The File Allocation Table has one entry per cluster. This entry uses 12, 16 or 28 bits for FAT12, FAT16 and FAT32.

The first two FAT entries

The first cluster of the data area is cluster #2. That leaves the first two entries of the FAT unused. In the first byte of the first entry a copy of the media descriptor is stored. The remaining bits of this entry are 1. In the second

entry the end-of-file marker is stored. The high order two bits of the second entry are sometimes, in the case of FAT16 and FAT32, used for dirty volume management: high order bit 1: last shutdown was clean; next highest bit 1: during the previous mount no disk I/O errors were detected.

(Historically this description has things backwards: DOS 1.0 did not have a BIOS Parameter Block, and the distinction between single-sided and double-sided 5.25" 360K floppies was indicated by the first byte in the FAT. DOS 2.0 introduced the BPB with media descriptor byte.)

FAT12

Since 12 bits is not an integral number of bytes, we have to specify how these are arranged. Two FAT12 entries are stored into three bytes; if these bytes are uv,wx,yz then the entries are xuv and yzw.

Possible values for a FAT12 entry are: 000: free, 002-fef: cluster in use; the value given is the number of the next cluster in the file, ff0-ff6: reserved, ff7: bad cluster, ff8-fff: cluster in use, the last one in this file. Since the first cluster in the data area is numbered 2, the value 001 does not occur.

DOS 1.0 and 2.0 used FAT12. The maximum possible size of a FAT12 filesystem (volume) was 8 MB (4086 clusters of at most 4 sectors each).

FAT16

DOS 3.0 introduced FAT16. Everything very much like FAT12, only the FAT entries are now 16 bit. Possible values for FAT16 are: 0000: free, 0002-ffef: cluster in use; the value given is the number of the next cluster in the file, fff0-fff6: reserved, fff7: bad cluster, fff8-ffff: cluster in use, the last one in this file.

Now the maximum volume size was 32 MB, mostly since DOS 3.0 used 16-bit sector numbers. This was fixed in DOS 4.0 that uses 32-bit sector numbers. Now the maximum possible size of a FAT16 volume is 2 GB (65526 clusters of at most 64 sectors each).

FAT32

FAT32 was introduced in Windows 95 OSR 2. It is not supported by Windows NT. Everything very much like FAT16, only the FAT entries are now 32 bits of which the top 4 bits are reserved. The bottom 28 bits have meanings similar to those for FAT12, FAT16. For FAT32: Cluster size used: 4096-32768 bytes.

Microsoft operating systems use the following rule to distinguish between FAT12, FAT16 and FAT32. First, compute the number of clusters in the data area (by taking the total number of sectors, subtracting the space for reserved sectors, FATs and root directory, and dividing, rounding down, by the number of sectors in a cluster). If the result is less that 4085 we have FAT12. Otherwise, if it is less than 65525 we have FAT16. Otherwise FAT32.

The maximum file size on a FAT32 filesystem is one byte less than 4 GiB (4294967295 bytes).

Microsoft operating systems use fff, ffff, xffffff as end-of-clusterchain markers, but various common utilities may use different values. Linux has always used ff8, fff8, but it now appears that some MP3 players fail to work unless fff etc. is used. Since 2.5.40 this is also what Linux uses.

1.4 Directory Entry

An example (6 entries on the same MSDOS floppy):

```
0009728 49 4f 20 20 20 20 20 20 53 59 53 27 00 00 00 00  IO      .SYS
0009744 00 00 00 00 00 00 08 5d 62 1b 1d 00 16 9f 00 00
0009760 4d 53 44 4f 53 20 20 20 53 59 53 27 00 00 00 00  MSDOS   .SYS
0009776 00 00 00 00 00 00 08 5d 62 1b 6d 00 38 95 00 00
0009792 43 4f 4d 4d 41 4e 44 20 43 4f 4d 20 00 00 00 00  COMMAND .COM
0009808 00 00 00 00 00 00 07 5d 62 1b b8 00 39 dd 00 00
0009824 44 42 4c 53 50 41 43 45 42 49 4e 27 00 00 00 00  DBLSPACE.BIN
0009840 00 00 00 00 00 00 08 5d 62 1b 27 01 f6 fc 00 00
0009856 4d 53 44 4f 53 20 20 20 20 20 20 28 00 00 00 00  MSDOS
0009872 00 00 00 00 00 00 1a 88 99 1c 00 00 00 00 00 00
0009888 46 44 49 53 4b 20 20 20 45 58 45 20 00 00 00 00  FDISK   .EXE
0009904 00 00 00 00 00 00 36 59 62 1b 02 00 17 73 00 00
```

```
Bytes   Content
0-10    File name (8 bytes) with extension (3 bytes)
11      Attribute - a bitvector. Bit 0: read only. Bit 1: hidden.
        Bit 2: system file. Bit 3: volume label. Bit 4: subdirectory.
        Bit 5: archive. Bits 6-7: unused.
12-21   Reserved (see below)
22-23   Time (5/6/5 bits, for hour/minutes/doubleseconds)
24-25   Date (7/4/5 bits, for year-since-1980/month/day)
26-27   Starting cluster (0 for an empty file)
28-31   Filesize in bytes
```

We see that the fifth entry in the example above is the volume label, while the other entries are actual files.

The "archive" bit is set when the file is created or modified. It is cleared by backup utilities. This allows one to do incremental backups.

As a special kludge to allow undeleting files that were deleted by mistake, the DEL command will replace the first byte of the name by 0xe5 to signify "deleted". As an extraspecial kludge, the first byte 0x05 in a directory entry means that the real name starts with 0xe5.

The first byte of a name must not be 0x20 (space). Short names or extensions are padded with spaces. Special ASCII characters 0x22 ("), 0x2a (*), 0x2b (+), 0x2c (,), 0x2e (.), 0x2f (/), 0x3a (:), 0x3b (;), 0x3c (<), 0x3d (=), 0x3e (>), 0x3f (?), 0x5b ([), 0x5c (\), 0x5d (]), 0x7c (|) are not allowed.

The first byte 0 in a directory entry means that the directory ends here.

Subdirectories start with entries for . and .., but the root directory does not have those.

VFAT

In Windows 95 a variation was introduced: VFAT. VFAT (Virtual FAT) is FAT together with long filenames (LFN), that can be up to 255 bytes long. The implementation is an ugly hack. These long filenames are stored in special directory entries. A special entry looks like this:

```
Bytes   Content
0       Bits 0-4: sequence number; bit 6: final part of name
```

```
1-10    Unicode characters 1-5
11      Attribute: 0xf
12      Type: 0
13      Checksum of short name
14-25   Unicode characters 6-11
26-27   Starting cluster: 0
28-31   Unicode characters 12-13
```

These special entries should not confuse old programs, since they get the 0xf (read only / hidden / system / volume label) attribute combination that should make sure that all old programs will ignore them.

The long name is stored in the special entries, starting with the tail end. The Unicode characters are of course stored little endian. The sequence numbers are ascending, starting with 1.

Now an ordinary directory entry follows these special entries, and it has the usual info (file size, starting cluster, etc.), and a short version of the long name. Also the unused space in directory entries is used now: bytes 13-17: creation date and time (byte 13: centiseconds 0-199, bytes 14-15: time as above, bytes 16-17: date as above), bytes 18-19: date of last access. (And byte 12 is reserved for Windows NT - it indicates whether the filename is in upper or in lower case; bytes 20-21 are reserved for OS/2: it stores there the index in EA DATA.SF, the file containing the extended attributes of all files. CYGWIN also uses a file EA DATA.SF.)

Old programs might change directories in ways that can separate the special entries from the ordinary one. To guard against that the special entries have in byte 13 a checksum of the short name:

```
int i; unsigned char sum = 0;

    for (i = 0; i < 11; i++) {
        sum = (sum >> 1) + ((sum & 1) << 7);  /* rotate */
        sum += name[i];                       /* add next name byte */
    }
```

An additional check is given by the sequence number field. It numbers the special entries belonging to a single LFN 1, 2, ... where the last entry has bit 6 set.

The short name is derived from the long name as follows: The extension is the extension of the long name, truncated to length at most three. The first six bytes of the short name equal the first six nonspace bytes of the long name, but bytes +,;=[], that are not allowed under DOS, are replaced by underscore. Lower case is converted to upper case. The final two (or more, up to seven, if necessary) bytes become ~1, or, if that exists already, ~2, etc., up to ~999999.

VFAT is used in the same way on each of FAT12, FAT16, FAT32.

1.5 FSInfo sector

FAT32 stores extra information in the FSInfo sector, usually sector 1.

```
Bytes   Content
0-3     0x41615252 - the FSInfo signature
4-483   Reserved
484-487 0x61417272 - a second FSInfo signature
488-491 Free cluster count or 0xffffffff (may be incorrect)
492-495 Next free cluster or 0xffffffff (hint only)
```

```
496-507 Reserved
508-511 0xaa550000 - sector signature
```

Next Previous Contents