Digital Forensics

Federico Conti

2024/25

Contents

The FAT File System Family

The FAT (File Allocation Table) File System is one of the earliest and simplest file systems, first developed in 1977/1978. Over time, it evolved into three main versions: FAT12, FAT16 and FAT32.

The number indicates the # of bits used to identify clusters

- FAT12 can address 212 = 4096 clusters. Windows permits cluster sizes from 512 bytes to 8 KB, which limits FAT12 to 32 MB
- FAT16 can address 216 = 65, 536 clusters
- FAT32 can address 228 clusters (top 4 bits used for other purposes)

Actually, first 2 & last 16 are reserved: usable clusters are slightly less.

Uses the MSDOS 8.3 filename format – Only 8 characters for the name + 3-character file extension (e.g., FILE1234.TXT).

VFAT (Virtual FAT) extends FAT to support long filenames with Unicode, maintaining backward compatibility.

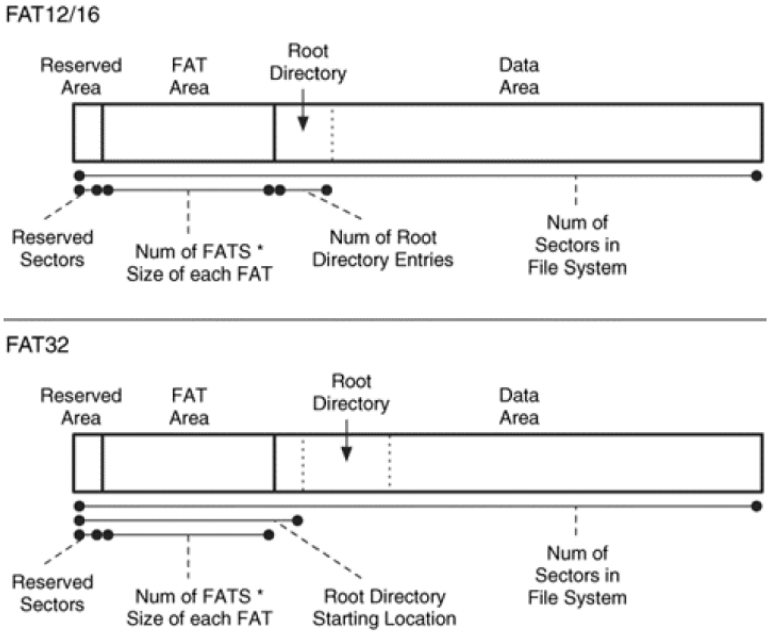
File sizes are stored as 32-bit integers, meaning the largest file size FAT32 can handle is 4 GB.

File sizes are stored as 32-bit integers.

Volume Organization

In FAT file systems, the storage device is divided into specific regions, each serving a defined role:

- The Volume Boot Record (VBR) contains the so-called BIOS Parameter Block
- The root directory of FAT12/16 has a fixed location and size
- FAT32 boot sector includes the locations of the root directory, FSINFO structure (that keeps track of free clusters, to optimize allocations), and boot-sector backup (should be 6)


FAT12/16
Reserved Area
FAT Area
Root Directory
Data Area
Reserved Sectors
Num of FATS * Size of each FAT
Num of Root Directory Entries
Num of Sectors in File System
FAT32
Reserved Area
FAT Area
Root Directory
Data Area
Reserved Sectors
Num of FATS * Size of each FAT
Root Directory Starting Location
Num of Sectors in File System

1. Reserved Area

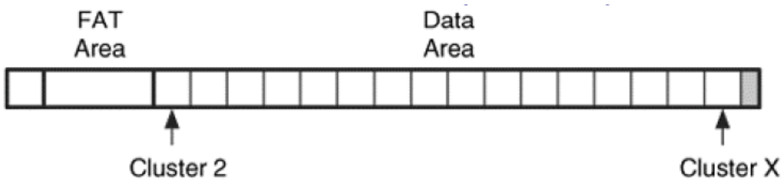
- Starts at sector 0.
- Contains the Volume Boot Record (VBR or Boot sector), which holds key information about the file system.
- FAT12/16: Usually 1 sector (only the VBR); FAT32: Larger because it includes FSINFO structure (helps track free clusters).

2. FAT Area

- follows the reserved area, and its size is calculated by multiplying the number of tables by their size

3. Data Area

- Clusters are only in Data Area, numbered from 2 (!!!) and after the root directory for FAT12/16
- Data could be also hidden after the last valid entry in a FAT table



The “Small Sectors” and “Large Sectors” fields represent the total number of sectors in the volume. Only one of these fields is used, and the other is set to zero.

Green (BIOS Parameter Block - BPB):

- Essential fields required for the basic operation of the file system.
- Defines sector sizes, cluster sizes, and disk structure.

Yellow (Extended BIOS Parameter Block - EBPB):

- Additional metadata introduced in later FAT versions.
- Includes details like the Volume Serial Number, Boot Signature, and Volume Label.

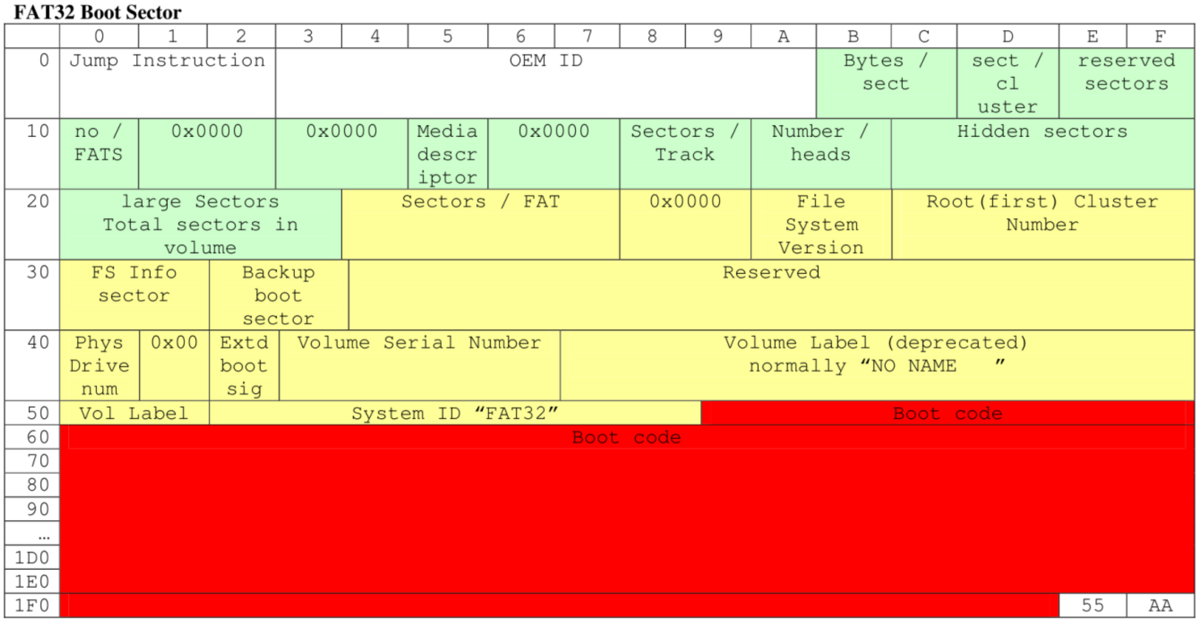
Boot sector FAT12/16

FAT16 Boot Sector

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Jump Instruction			OEM ID								Bytes / sect		sect / cluster	reserved sectors	
10	no / FATS	Root entries		Small Sectors		Media descriptor	Sectors / FAT		Sectors / Track		Number / heads		Hidden sectors			
20	large Sectors				Physical drive number	reserved	ext boot sig	Volume Serial Number				Volume Label (deprecated)				
30	Volume label						File System Type									
40	OS Boot Code															
50																
60																
70																
…																
1D0																
1E0																
1F0															55	AA

Boot sector FAT32

FAT32 Boot Sector



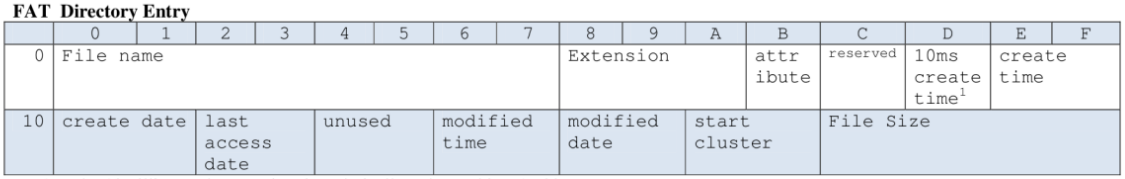
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Jump Instruction			OEM ID								Bytes / sect		sect / cl uster	reserved sectors	
10	no / FATS	0x0000		0x0000		Media descr iptor	0x0000		Sectors / Track		Number / heads		Hidden sectors			
20	large Sectors Total sectors in volume				Sectors / FAT				0x0000		File System Version		Root(first) Cluster Number			
30	FS Info sector		Backup boot sector		Reserved											
40	Phys Drive num	0x00	Extd boot sig	Volume Serial Number				Volume Label (deprecated) normally “NO NAME ”								
50	Vol Label		System ID “FAT32”							Boot code						
60	Boot code															
70																
80																
90																
…																
1D0																
1E0																
1F0															55	AA

https://www.writeblocked.org/resources/FAT_cheatsheet.pdf

Files and Directories

A directory entry in FAT file systems is a 32-byte record that stores metadata about a file or directory.

FAT Directory Entry

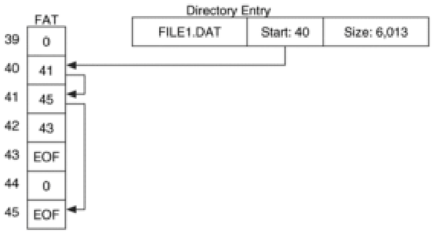


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	File name								Extension			attr ibute	reserved	10ms create time[1]	create time	
10	create date		last access date		unused		modified time		modified date		start cluster		File Size			

[1] The 10millisecond create time is technically only used in FAT32.

The File Allocation Table (FAT) keeps track of file storage using cluster chains. Each file’s data is stored in clusters, and the FAT table links these clusters together to form a chain.

- Each FAT entry points to the next cluster in the file’s.
- Clusters marked EOF (End of File) indicate the last cluster of a file.



`fsstat` decodes this chains (in sectors); special values:

```
0 → not allocated
0xf...ff0-0xf...f6 → reserved
0xf...ff7 → damaged
0xf...ff8-0xf...fff → EOF
```

Note: FAT entries start at 0, but:

- The first addressable cluster #2.

- Entry 0 typically stores a copy of the media type, and entry 1 stores the dirty-status of the file system

Example

In eighties.dd (SHA256: cc121c3a…) and eighties-all-files.dd (SHA256:e5f16884…) you'll find two very similar FAT16 (not VFAT) file systems. In the former all files have been deleted. Using ImHex.

1. find out: Sector and cluster sizes Number of reserved sectors Locations of: FAT1, FAT2, Root Dir. (=Data Area), first cluster (#2)

compare these results with the output of fsstat

2. check the FAT entries for 48.gif in the two dd-images, and compare the results of istat on "inode" 5

```
fls -r eighties-all-files.dd
  ##OUT##
  d/d 3:  jpgs
  + r/r 517:      clive.jpg
  + r/r 518:      48k.jpg
  d/d 4:  games
  + r/r 581:      mmonty.tzx
  r/r 5:  48.gif
  r/r 6:  48.txt
  v/v 523203:     $MBR
  v/v 523204:     $FAT1
  v/v 523205:     $FAT2
  V/V 523206:     $OrphanFiles
  ###

istat eighties-all-files.dd 5
  ##OUT##
  Sectors:
  108 109 110 111 116 0 0 0 #l'ha recuperato dentro la FAT
  ###

fsstat eighties-all-files.dd

  ##OUT##
  FAT CONTENTS (in sectors)

  100-103 (4) -> EOF
  104-107 (4) -> EOF
  108-111 (4) -> 116 # cluster chain
  112-115 (4) -> EOF
  116-119 (4) -> EOF
  120-331 (212) -> EOF
  332-1211 (880) -> EOF
  1212-1279 (68) -> EOF
  ###
```

When a file name exceeds the 8.3 format, the file system creates additional directory entries to store the name in Unicode (2 bytes per character).

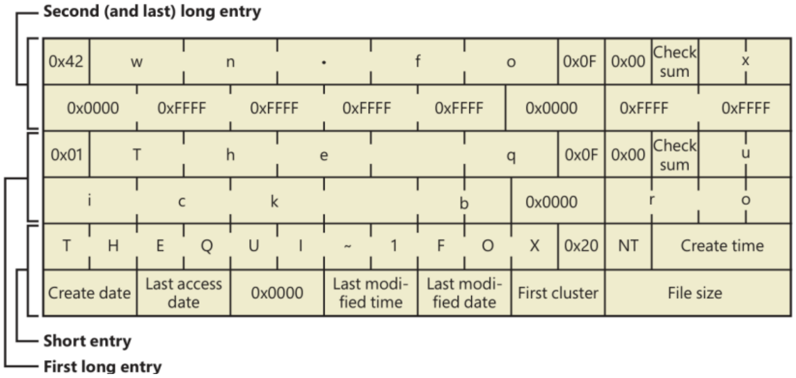
These LFN entries are linked together and precede the main directory entry (which still stores the short 8.3 name for compatibility).

- Each LFN entry is marked with the attribute 0x0F, meaning it is not treated as a normal file entry.
- The last LFN entry in the sequence has its sequence number OR-ed with 0x40; or 0xe5 if unallocated.

Long File Name

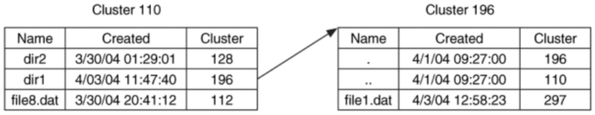
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	file name (Unicode 2 bytes/char)											0x0F	reserved	Check sum	file name	
10	file name										0x0000		file name			

The quick brown.fox”, as THEQUI~1.FOX in 8.3 convention.



When a new directory is created, it contains

. and ..

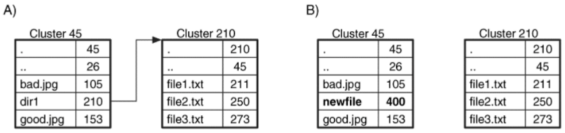

Cluster 110

Name	Created	Cluster
dir2	3/30/04 01:29:01	128
dir1	4/03/04 11:47:40	196
file8.dat	3/30/04 20:41:12	112

Cluster 196

Name	Created	Cluster
.	4/1/04 09:27:00	196
..	4/1/04 09:27:00	110
file1.dat	4/3/04 12:58:23	297

those entries can be helpful for carving deleted directories


A)

Cluster 45

.	45
..	26
bad.jpg	105
dir1	210
good.jpg	153

Cluster 210

.	210
..	45
file1.txt	211
file2.txt	250
file3.txt	273

B)

Cluster 45

.	45
..	26
bad.jpg	105
newfile	**400**
good.jpg	153

Cluster 210

.	210
..	45
file1.txt	211
file2.txt	250
file3.txt	273

Since the size of a directory is always 0, the only way to know how many cluster to read is following the cluster chain

Example

In eighties-vfat.dd (SHA256: 62258f92ebb42226…) and eighties-vfat-all-files.dd (SHA256: fe46141b98d227cb…) you’ll find two very similar, and familiar, VFAT FAT16 file systems. As with the previous exercise, in the former all files have been deleted.

Yet, fls -rp eighties-vfat.dd can show the full, long name, for some deleted files but not for others, that are listed under $OrphanFiles.

1. Can you explain why? Hint: eighties-vfat-all-files.dd contains some clues

In some cases, even if the file is cacelled, we have the full name, and it is strange because in the fat one byte '_' is put above the first character (eighties.dd). Since we have a vfat there are the entries with the long name and we can trace the original name. OrphanFiles is a standrd used by TSK when it does not have babstanz ainomraizons to know where that file is.

2. Using ImHex, can you manually recover the full names from eighties-vfat.dd?

```
fls -rp eighties-vfat.dd
  ##OUT##
  r/r * 3:        _
  r/r * 4:        _
  d/d * 6:        Games
  r/r * 583:      Games/Mutant Monty.tzx
  r/r * 7:        _8.gif
  r/r * 8:        _8.txt
  v/v 523203:     $MBR
  v/v 523204:     $FAT1
  v/v 523205:     $FAT2
  V/V 523206:     $OrphanFiles
  -/r * 519:      $OrphanFiles/_LIVES~1.JPG
  -/r * 520:      $OrphanFiles/_8k.jpg
  ###
```