SYSTEM CALL MAGAZINE

ISSUE 1 - MAY 2008

In this issue:

Editor’s Comments:

The Past:

....

Understanding the FAT File System

by somebody

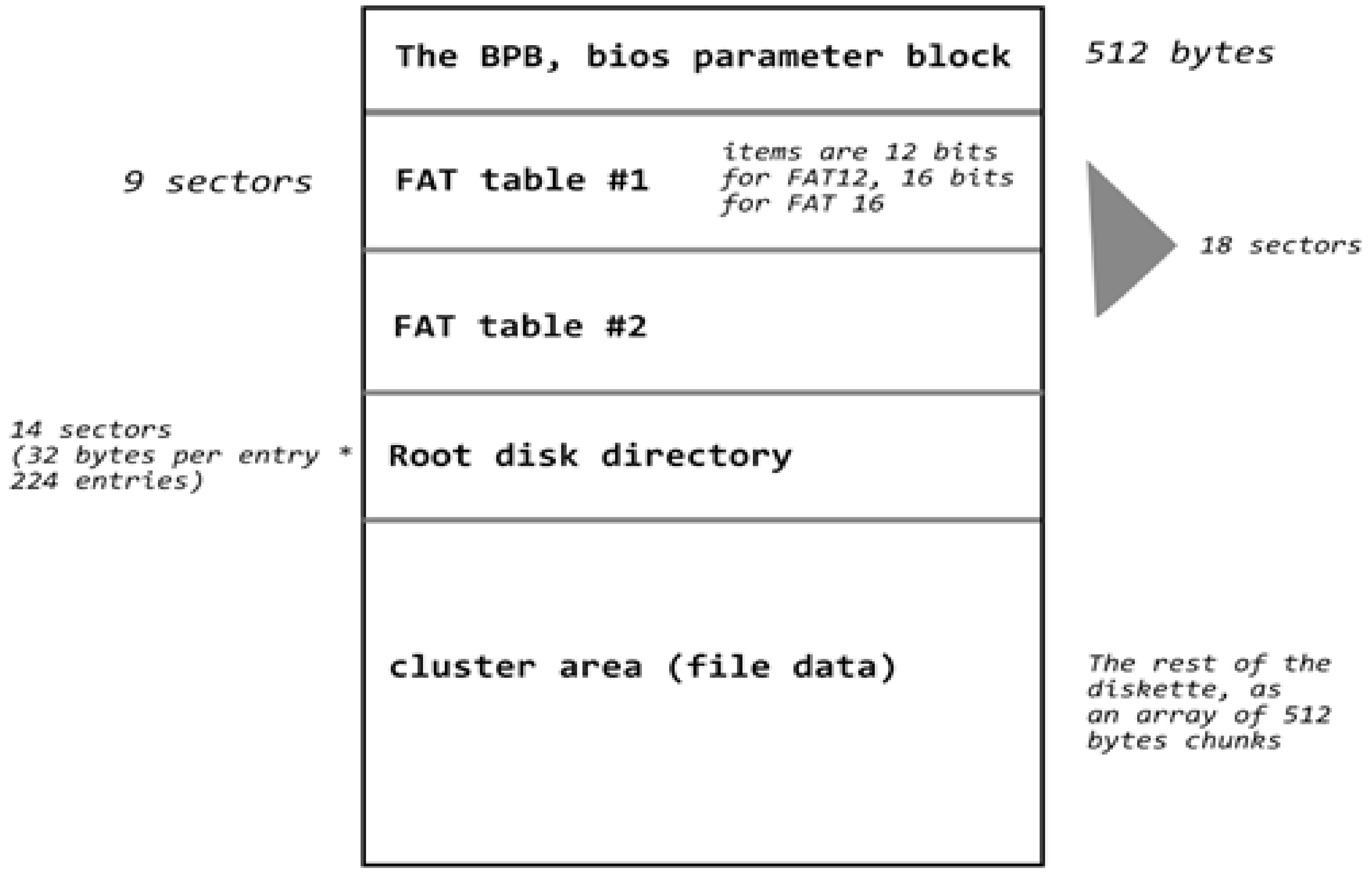
The FAT file system was designed to manage disks in Stand-Alone Disk BASIC by Marc McDonald, with help from Bill Gates in 1977. Over the time, the FAT file system has been updated to allow for bigger partitions. Our primary focus today is FAT12 as this is use in most hobbyist operating systems.

The FAT file system has three sections (in order they appear on disk):

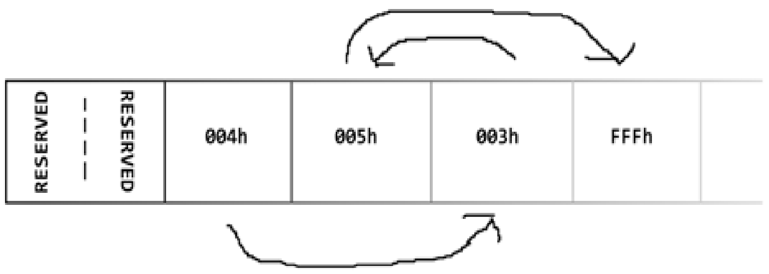
1. The Boot Sector (contains the BPB and code to start the operating system, if one exists on the file system)
2. The File Allocation tables
3. Cluster Area

The boot sector is the most important part of the file system and it is only 512 bytes long! This is because often contains a boot loader. It begins with the BPB which defines important fields describing the file system structure. These fields include include bytes per sector, sectors per cluster, reserved sector count and sectors per file allocation table.

Following the boot sector is the File allocation tables (FAT). The FAT keep track of which clusters are used by files and in which order. Data storage is not perfect, and as such it also keep track of which clusters are bad. The FAT consists of an array of pointers. For FAT12, it's an array of 12-bit numbers pointing to next cluster within the file.


The BPB, bios parameter block
512 bytes
9 sectors
FAT table #1
items are 12 bits for FAT12, 16 bits for FAT 16
18 sectors
FAT table #2
14 sectors (32 bytes per entry * 224 entries)
Root disk directory
cluster area (file data)
The rest of the diskette, as an array of 512 bytes chunks

The cluster area is where the file data is stored. Sectors are grouped together into clusters. The number of sectors in each cluster is defined in the BPB, which is usually a single sector for floppy disks.
Directories are also located in the cluster area, which are files that contain an array of directory entries. Each directory entry contains the name, size, location of the first cluster, and attributes of files within the that directory.
To find a file, the file system looks at the root directory(which is located in the second cluster). Knowing where the first cluster of a file is, we can then read the entire file (or directory).



Offset	**Length**	**Description**
0x00	3	Jump instruction. This instruction will be executed and will skip past the rest of the (non-executable) header if the partition is booted from. See Volume Boot Record. If the jump is two-byte near jmp it is followed by a NOP instruction.
0x03	8	OEM Name (padded with spaces). MS-DOS checks this field to determine which other parts of the boot record can be relied on. Common values are IBM 3.3 (with two spaces between the “IBM” and the “3.3”), MSDOS5.0 and MSWIN4.1.
0x0b	2	Bytes per sector. A common value is 512, especially for file systems on IDE (or compatible) disks. The BIOS Parameter Block starts here.
0x0d	1	Sectors per cluster. Allowed values are powers of two from 1 to 128. However, the value must not be such that the number of bytes per cluster becomes greater than 32 KiB.
0x0e	2	Reserved sector count. The number of sectors before the first FAT in the file system image. Should be 1 for FAT12/FAT16. Usually 32 for FAT32.
0x10	1	Number of file allocation tables. Almost always 2.
0x11	2	Maximum number of root directory entries. Only used on FAT12 and FAT16, where the root directory is handled specially. Should be 0 for FAT32. This value should always be such that the root directory ends on a sector boundary (i.e. such that its size becomes a multiple of the sector size). 224
0x13	2	is typical for floppy disks.
0x15	1	Total sectors (if zero, use 4 byte value at offset 0x20)
		Media descriptor

Reading a file

In order to read a file we must do the following:

1. Load the FAT.
2. Find the root directory (which is located at second cluster).
3. Find the file entry on the directory.
4. Get the pointer to the first cluster of the file, in the directories entry.
5. Go to the cluster pointed to.
6. Read cluster.
7. Go to the cluster's FAT entry. And read the pointer.
8. If pointer is not equal to 0xFFF (end of file) go to step 3.