FAT File System

The so-called FAT file system, which is the file system used in all versions of the MS-DOS operating system to date and in the first two releases of OS/2' (Versions 1.0 and 1.1), has a dual heritage in Microsoft's earliest programming language products and the Digital Research CP/M operating system--software originally written for 8080-based and Z-80-based microcomputers. It inherited characteristics from both ancestors that have progressively turned into handicaps in this new era of multitasking, protected mode, virtual memory, and huge fixed disks.

The FAT file system revolves around the File Allocation Table for which it is named. Each logical volume has its own FAT, which serves two important functions: it contains the allocation information for each file on the volume in the form of linked lists of allocation units (clusters, which are power-of-2 multiples of sectors) and it indicates which allocation units are free for assignment to a file that is being created or extended.

Microsoft Corporation, 1978

The FAT was invented by Bill Gates and Marc McDonald (second row, second from right) in 1977 as a method of managing disk space in the NCR version of standalone Microsoft's Disk BASIC. Tim Paterson, at that time an employee of Seattle Computer Products (SCP), was introduced to the FAT concept when his company shared a booth with Microsoft at the National Computer Conference in 1979. Paterson subsequently incorporated FATs into the file system of 86-DOS, an operating system for SCP s S-100 bus 8086 CPU boards. 86-DOS was eventually purchased by Micro-soft and became the starting point for MS-DOS Version 1.0, which was released for the original lBM PC in August 1981.

When the FAT was conceived, it was an excellent solution to disk management, mainly because the floppy disks on which it was used were rarely larger than 1 Mb. On such disks, the FAT was small enough to be held in memory at all times, allowing very fast random access to any part of any file. This proved far superior to the CP/M method of tracking disk space, in which the information about the sectors assigned to a file might be spread across many directory entries, which were in turn scattered randomly throughout the disk directory. When applied to fixed disks, however, the FAT began to look more like a bug than a feature. It became too large to be held entirely resident and had to be paged into memory in pieces: this paging resulted in many superfluous disk head movements as a program was reading through a file and degraded system throughput. In addition, because the information about free disk space was dispersed across many sectors of FAT, it was impractical to allocate file space contiguously, and file fragmentation became another obstacle to good performance. Moreover, the use of relatively large clusters on fixed disks resulted in a lot of dead space, since an average of one-half cluster was wasted for each file. (Some network servers use clusters as large as 64Kb.)

The FAT file system 's restrictions on naming files and directories are inherited from CP/M. When Paterson was writing 86-DOS one of his primary objectives was to make programs easy to port from CP/M to his new operating system. He therefore adopted CP/M's limits on filenames and extensions so the critical fields of 86-DOS File Control Blocks (FCBs) would look almost exactly like those of CP/M. The sizes of the FCB filename and extension fields were also propagated into the structure of disk directory entries. In due time 86-DOS became MS- DOS and application programs for MS-DOS proliferated beyond anyone's wildest dreams. Since most of the early programs depended on the structure of FCBs the 8.3 format for filenames became irrevocably locked into the system.

During the last couple of years Microsoft and IBM have made valiant attempts to prolong the useful life of the FAT file system by lifting the restrictions on volume sizes improving allocation strategies caching path names and moving tables and buffers into expanded memory. But these can only be regarded as temporizing measures because the fundamental data structures used by the FAT file system are simply not well suited to large random access devices. The HPFS solves the FAT file system problems mentioned here and many others but it is not derived in any way from the FAT file system. The architect of the HPFS started with a clean sheet of paper and designed a file system that can take full advantage of a multitasking environment and that will be able to cope with any sort of disk device likely to arrive on microcomputers during the next decade.

Source: www.seds.org