FAT32 vs. NTFS

Jason Capriotti
CS384, Section 1
Winter 1999-2000
Dr. Barnicki
January 28, 2000

Table of Contents

List of Figures

Introduction

Since they were developed over twenty years ago, the role of personal computers in our lives has drastically increased. Businesses as well as individuals use computers to store and manipulate data, produce products, and conduct business. However, when people turn on their computers and conduct their business, they probably don’t care too much about the constant reading and writing of data on the computer’s hard drive.

Over the years, hard drives and the systems used to store data on them have constantly evolved. While the two latest Windows file systems, FAT32 and NTFS, both look similar on the outside, they are really quite different, and have several advantages and disadvantages.

The Physical Disk

Before file systems can be understood, the physical disk itself must also be understood. The type of physical disk that most people are used to, is a hard drive. Also known as fixed disks, they are where computers store their operating systems, program files, and data.

Hard drives contain several platters, which are made up of either aluminum or more recently, glass. Glass is used in the newest and largest drives, because the platters can be made thinner, and they are more resistant to the heat that the drives generate.

In order for the platters to store information, they are coated with a magnetically sensitive material, primarily containing iron oxide and cobalt alloy.

The data on the platters is accessed by a head, which is moved by the head arm. There is usually one head per platter side. Figure 1 shows the primary components of a basic hard drive.

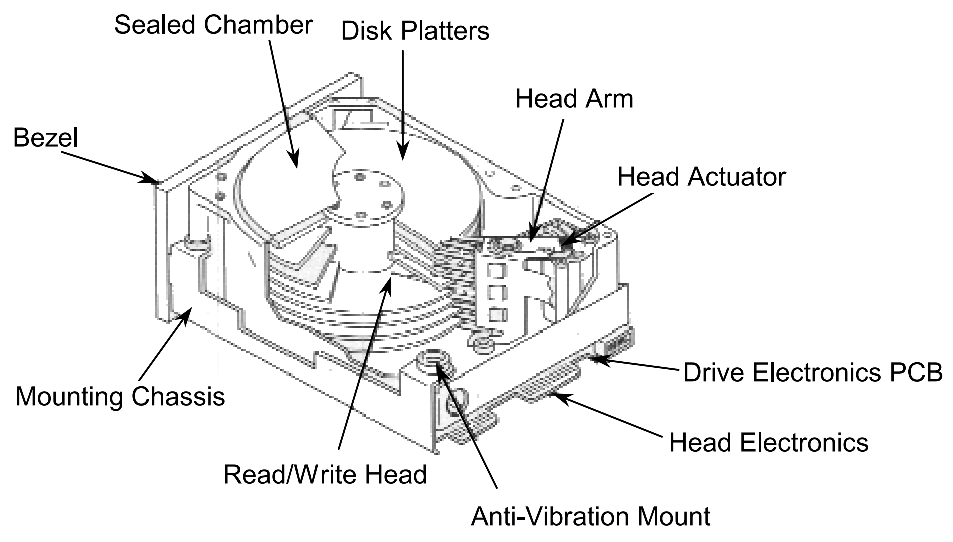

Figure 1 - Components of a hard disk drive assembly ("Hard Drives")

Information stored on the hard drive is divided up into many distinct areas on the platters. Tracks, sectors, and cylinders are what define these areas.

A track is a ring around a platter, containing information. Tracks on each platter line up with each track on the platter above or below. Each set of tracks that are identically positioned above or below each other are called cylinders.

The final way that the information in hard drives is divided is into sectors. Sectors on a platter can be thought of as slices of a pie. Sectors are necessary, because there would otherwise be too much information in each track (“Hard Drives”).

File System Basics

Operating systems need a way to manage the data that is stored on a hard drive. This is where file systems come into play. Just as people organize their CD or DVD collections for retrieval at a later date, operating systems need a way to organize files for retrieval.

File Attributes

A file can be thought of as an abstract data type. The attributes that most operating systems would give to a file include the following:

- **Name** – This is the attribute of the file that computer users are used to seeing on a regular basis. It is simply the symbolic name of the file.
- **Type** – The type of a file defines what operations can be performed on it. Types of files include executable, text, image, etc. The operating system usually associates a certain action with each type of file (e.g. open with Microsoft Word, or execute).
- **Location** – This is a pointer to the file on the device on which it resides (e.g. a hard disk drive).
- **Size** – This is the size of the file, usually in bytes, words, or blocks.

- **Protection** – This is usually specified by a user, and defines the level of access that other users can have. Different levels may include reading, executing, writings, etc.
- **Time, date, and owner** – Most operating systems usually keep track of a file's creation, last modification, and last use date and time. Certain operating systems (such as Windows NT) also keep track of who created the file (Silberschatz 338).

File Operations

As an abstract data type, there are several different operations that can be performed on a file. These operations are provided by the operating system as system calls. The basic file operations include:

- **Creating** – In order to create a file, the operating system must allocate the proper amount of storage space. Then, an entry into the file's directory must be made, in order for a user to see that the file is in that directory.
- **Writing** – To write to a file, the operating system controls a write pointer, which specifies the next location to write to. Whenever a write occurs, this pointer must be updated.
- **Reading** – Much like writing, the operating system controls a read pointer, which specifies the next location to read from. Once a read occurs, the read pointer is updated.

- **Deleting** – The operating system first locates the proper file to delete. Once it is found, the space associated with the file is freed, and its entry in the directory is removed (Silberschatz 339).

Tree Directory Structure

A directory is what allows users to organize files according to their purpose. The most common directory structure is the tree structure. In this type of structure, there is a root directory, which contains files and subdirectories. The root's subdirectories can contain other directories, and files. These subdirectories can contain other subdirectories, and so on. Figure 2 shows what a tree directory structure looks like.

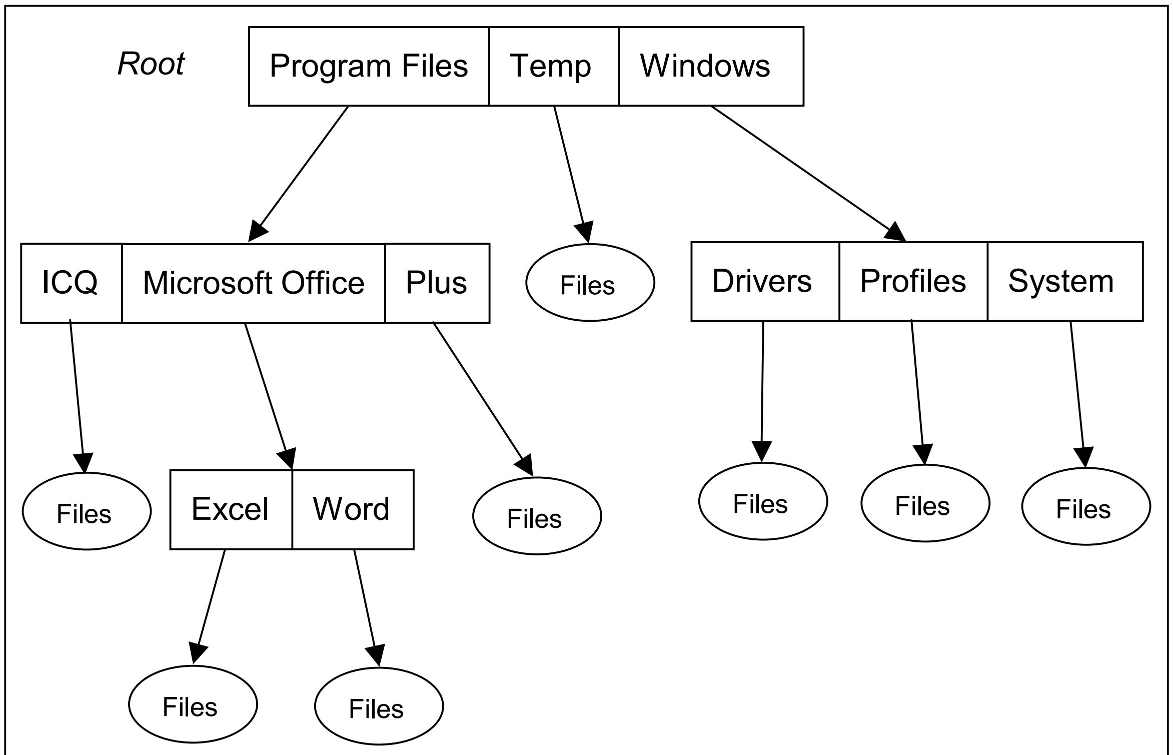

Figure 2 – Tree directory structure (Silberschatz)

The FAT32 File System

FAT32 is the latest version of the FAT (file-allocation table) file system. FAT32 was developed by Microsoft to accommodate the ever-increasing size of hard disk drives. It was first introduced with Windows 95 OSR2, and later with Windows 98, and Windows 2000.

Origins of FAT32

In order to explain how FAT32 operates today, the origins of the FAT file system must be explored.

FAT originated with MS-DOS in 1977. At the time, DOS was a 16-bit operating system, built around 16-bit computer architecture. Therefore, being a 16-bit operating system, each sector of a hard disk drive was assigned a 16-bit number, limiting the number of sectors to 65,535. With 512 byte sectors, the maximum size of a hard drive could be 32MB. If someone had a larger hard drive, it would have to be partitioned into sections equal to or less than 32MB.

With the release of MS-DOS 4.0, there was a solution to the 32MB limit. Sectors were grouped into clusters. Now, instead of storing sector addresses in the FAT, cluster addresses would be stored. With this enhancement, FAT could now handle 65,535 clusters, rather than sectors, and thus could handle larger hard drives. Figure 3 shows how the operating system determines cluster size.

Partition Size	Cluster Size
0 MB – 32 MB	0.5 KB
33 MB – 64 MB	1 KB
65 MB – 128 MB	2 KB
129 MB – 256 MB	4 KB
257 MB – 512 MB	8 KB
513 MB – 1 GB	16 KB
1 GB – 2 GB	32 KB
2.1 GB – 4 GB	64 KB

Figure 3 – How partition size determines cluster size in FAT ("Examining")

As shown in Figure 3, cluster sizes drastically increase once hard drives reach 1GB. This leads to a problem, since sections of files are stored in entire clusters. Therefore, if a 40KB file was store on a drive with 32KB cluster sizes, it uses 2 clusters, or 64KB of space. While wasting 24KB of space may not seem like much, this wasted space can add up on larger drives.

Once again, the FAT system was enhanced to accommodate even larger disk drives. This latest version is FAT32 uses smaller cluster sizes, in order to decrease the amount of wasted space. For drives under 8GB, the cluster size is 4KB. Since FAT32 is based on the original FAT system, it is still compatible with existing components that require FAT ("Examining").

Disk Structure

The basis of the FAT32 file system is the FAT itself. A portion of the disk or partition is set aside by the operating system to store the FAT. The FAT is basically a linked list, with one entry for each block of space (cluster) on the disk. A directory entry contains the number of the first block used by the file. The FAT entry at that block contains the number of the next block, and so on. The last block contains a special end-of-file value in the FAT. Figure 4 shows a simple FAT diagram (Silberschatz 378).

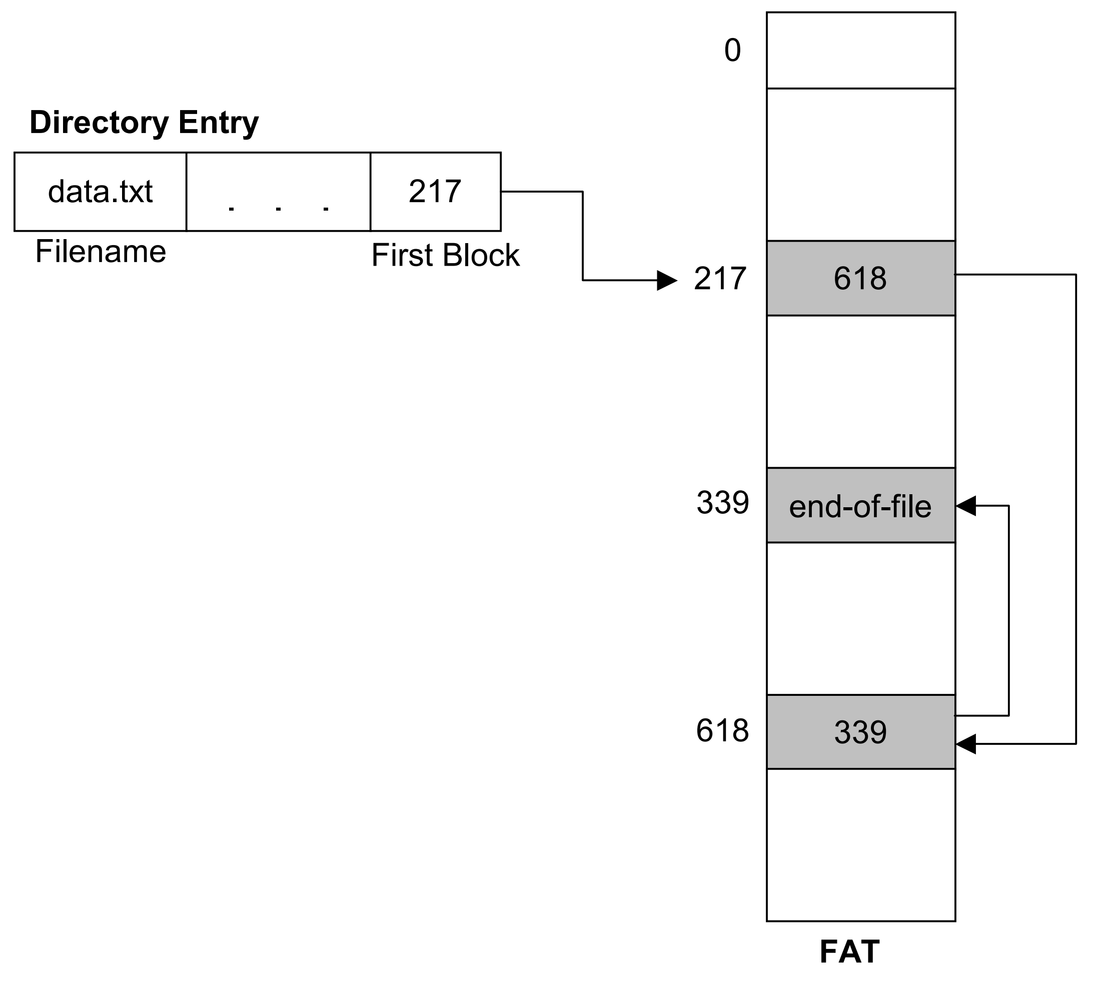

Figure 4 – File-allocation table (Silberschatz)

When files are accessed under the FAT system, the disk drive’s heads must be constantly repositioned between the FAT and the file itself. This can lead to decreased performance on large partitions (“Overview”).

Virtual FAT

Although Windows 95/98 supports long filenames, because FAT32 is still based off of its 1977 ancestor, it still uses the 8.3 (e.g. “12345678.123”) naming convention. For this reason, the Virtual FAT (VFAT) was developed.

VFAT really refers to a Windows driver, which runs in protected mode. It was developed by Microsoft as a driver, so that other operating systems could use it to access Windows 95/98 long filenames in FAT partitions. It can be thought of as an interpreter between FAT and a user. It interprets how a file looks to the FAT, to how the user sees the file (“What Is”).

Here is an example to explain how VFAT works. A user saves the file “Sales Report For Tammy.doc” to a disk drive. The filename in the FAT directory entry is “salesr~1.doc”, which is the first six characters of the filename, a tilde (~), and a number (starting with 1). If that file already exists, the number is simply incremented. VFAT then stores the long filename using additional directory entries immediately after the first directory entry. Each additional directory entry stores 13 characters of the long filename. Since long filenames support characters that the 8.3 naming convention does not, there will always be at least one of these additional directory entries made by the VFAT.

DOS ignores the additional directory entries made by VFAT, because of the entry's attribute field. This field is set to 0Fh, which allows it to be ignored by DOS (Buttron 17).

Benefits

Because of the new, smaller cluster size of FAT32, it has some clear benefits, instead of one major downfall.

One of the major benefits is that it is a more widely accepted file system. One area where this comes into play is on a multiboot computer. Computers with more than one bootable operating system usually have at least one FAT32 (or FAT) partition. This is because many operating systems such as DOS, Windows 3.x/95/98/NT, and OS/2 support the FAT system (Daily 97).

Another benefit of FAT32 is its low overhead on smaller partitions. The FAT32 file system is the best choice for partitions under 200MB. Because of its low overhead, Windows NT can only format floppy disks using the FAT (or FAT32 with Windows 2000) file system. This is because NTFS's overhead cannot fit onto a floppy disk.

Another benefit that DOS users may recognize is undelete utilities. These utilities directly access the system's hardware, which Windows NT does not allow. However, if the system could be restarted into DOS, any FAT/FAT32 partitions could be accessed by undelete utilities ("Overview").

The NTFS File System

Being a relatively new file system, NTFS was developed with the downfalls of others kept in mind. Microsoft intended it to be robust, secure, and fast. The key advantage that its developers had was that they were starting from scratch, and did not have to retain compatibility with a 20-year-old file system (Daily 98).

Much like FAT32, NTFS stores files in clusters of sectors on the hard drive. However, the cluster sizes that NTFS uses are much different from FAT32, as shown in Figure 5.

Partition Size	Cluster Size
0 MB – 512 MB	512 bytes
512 MB – 1 GB	1 KB
1 GB – 2 GB	2 KB
2 GB or more	4 KB

Figure 5 – NTFS cluster sizes (Daily)

NTFS clusters are addressed from the beginning of the partition to the end using logical cluster numbers (LCNs). Using an LCN address, the physical disk offset of a file can be calculated by multiplying the LCN by the cluster size (Silberschatz 766).

Master File Table

Each file in an NTFS partition has a unique ID (64-bit number) called a file reference. This number contains a 48-bit file number and a 16-bit sequence number. The sequence number is incremented every time the MFT entry is reused. It is used by NTFS for internal consistency checks. The full 64-bit number is the record number in the Master File Table (MFT) that describes the files. One or more records in the MFT describe a file's attributes (including security settings).

The MFT is actually a file that NTFS maps using records within the MFT. It has base pre-assigned base records, which are associated with disk management. These are known as metadata files. Figure 6 shows these entries in the MFT (Russinovich 63).

Name	Record	Description
$MFT	0	Master File Table
$MFTMIRR	1	Copy of the first 16 records of the MFT
$LOGFILE	2	Transactional logging file
$VOLUME	3	Contains volume serial number, creation time, and dirty flag
$ATTRDEF	4	Attribute definitions
.	5	Root directory of the disk
$BITMAP	6	Contains drive's cluster map (in-use vs. free)
$BOOT	7	Boot record of the drive
$BADCLUS	8	Lists bad clusters on the drive
$QUOTA	9	Contains user quota information – unused before NT 5.0 (Windows 2000) NTFS
$UPCASE	10	Maps lowercase characters to their uppercase version

Figure 6 – NTFS metadata files in the MFT (Russinovich)

MFT Records

Records in the MFT contain four basic fields. The first field is a header, which contains basic information about the record. This is followed by one or more attributes that describe the characteristics of the file or directory, which the MFT record represents. The header data includes the 16-bit sequence numbers , a pointer to the first attribute in the record, a pointer to the first free byte in the record, and the MFT record number. Figure 7 shows this layout (Russinovich 63).

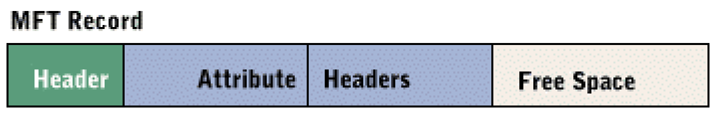

Figure 7 – MFT record structure (Russinovich)

Security

NTFS partitions provide high levels of security for its contents, because of its tight integration with NT security. File level security is provided by NTFS for setting permissions of files and folders. Windows NT then gets and sets this information for each user. When a user is logged on, the NTFS permissions determine what a user has access to, providing a very secure environment.

Another security feature is that an NTFS partition cannot be accessed if the system is booted from a DOS disk. NT requires users to login to the system using their username and password (Russinovich 64).

Reliability

When NTFS stores data to a disk, file I/O events are recorded to a special transaction log. The transactions in this log are not committed, until the action is successfully completed. Therefore, if a system crash or power interruption occurs, this log can be used to restore the partition, and prevent corruption.

Another feature is a background housekeeping process that automatically prevents data being written to bad disk sectors. Since this is done in the background, a process accessing the disk will never know that the bad sector exists (Russinovich 64).

Performance

One of the design goals of NTFS was speed. NTFS provides excellent performance on large disk partitions. In order for the performance to be adequate, however, the disk partition must be larger than 400MB, because of NTFS's relatively high overhead.

Another way performance is increased in NTFS is its binary tree structure for directories. This reduces the number of disk reads required to locate files. In addition to this, files in each directory are sorted on the fly, providing faster access to them as well.

In addition to NTFS's performance increases seen by its file storage techniques, its resistance to fragmentation is also a benefit. When NTFS writes to the disk, it makes intelligent choices about where to store the files (Russinovich 64).

Comparison of FAT32 and NTFS

After describing each files system, there are clear advantages and disadvantages to each system. The following is an overview of these advantages and disadvantages.

FAT32

Advantages

- Excellent performance on partitions under 200MB.
- Able to use undelete utilities.
- Compatible with many different operating systems, and is often used as the primary partition on multiboot systems.

Disadvantages

- Performance continues to decrease on partitions over 200MB.
- Unsecured
- Prone to fragmentation.
- Directory structure has no formal organization.

NTFS

Advantages

- Very secure.
- Excellent performance on partitions over 400MB.
- File and directory structure enhances performance.
- Reliable, less prone to corruption.

- Less prone to fragmentation.

Disadvantages

- Poor performance on partitions under 400MB.
- Not supported by most operating systems.

Conclusion

For users of Microsoft Windows based operating systems, there are two choices for the file system to use on a computer’s partitions: NTFS and FAT32. While FAT32 has its advantages, NTFS clearly has greater benefits. With the exception of a multiboot system, or a single boot system running Windows 9x, NTFS is the way to go.

Works Cited

Buttron, Jason. “Microsoft’s Original DOS File System, FAT12, and Its Evolution to Become FAT32, Windows 95’s Most Current File System.” Milwaukee School of Engineering, Design of Operating Systems. Spring 1997.

Daily, Sean. “NTFS vs. FAT.” *Windows NT Magazine* October 1996: 95.

“Examining the New FAT32 System.” *Inside Microsoft Windows 95* January 1998. Online. <http://proquest.umi.com>.

“Hard Drives.” *CircuitMasters*. 27 October 1998. <http://www.circuitmasters.com/tech/hdindex.html>.

“Overview of FAT, HPFS, and NTFS File Systems.” *Microsoft Knowledge Base* Article ID: Q100108. Online. <http://support.microsoft.com/support/kb/articles/Q100/1/08.asp>

Russinovich, Mark. “Inside NTFS.” Windows NT Magazine January 1998: 61.

Silberschatz, Abraham and Peter Baer Galvin. *Operating System Concepts*. New York: John Wiley & Sons, Inc, 1999.

“What Is… VFAT.” 26 November 1999. Online. <http://www.whatis.com/vfat.htm>.