v. 3, n. 15, 2023

Journal of
Engineering
Research

NTFS FILE SYSTEM: AN INTRODUCTORY ANALYSIS

José Gleisson da Costa Germano
PhD student in Teaching, with emphasis in Science, Mathematics and Engineering Teaching -Academic Doctorate Program in Teaching, RENOEN, Instituto Federal do Ceará -IFCE, Campus Fortaleza

José Wally Mendonça Menezes
Supervisor -Academic Doctorate Program in Teaching, RENOEN, with emphasis in Science, Mathematics and Engineering Teaching –''Instituto Federal do Ceará"-IFCE, Campus Fortaleza

All content in this magazine is licensed under a Creative Commons Attribution License. Attribution-Non-Commercial-Non-Derivatives 4.0 International (CC BY-NC-ND 4.0).

DOI 10.22533/at.ed.3173152308052

Abstract: This paper is intended to provide some introductory elucidations concerning file systems. It is part of an ongoing qualitative research in which a general concept of file system is presented, and the main functions and/or roles that a file system plays. Furthermore, the work exposes an approach about the structure and implementation of directories, about the allocation method, free space management, efficiency and performance and brings clarifications about retrieval and information stored in the FCB.

Keywords: File system; NTFS system; Introductory elucidations.

INTRODUCTION

The present work comes to light in order to provide certain elucidations concerning file systems. It is divided into two topics, 1 and 2. Topic 1, as it is noticeable, brings an introduction about the work. In item 1.1 there is a presentation of a general concept of file system; item 1.2 discusses the functions and/or the main roles of a file system. In topic 2, an approach is made about the structure, implementation and deployment of directories (in item 2.1), about allocation method, free space management, efficiency and performance and brings clarifications about recovery and information stored in FCB (in item 2.2).

GENERAL FILE SYSTEM CONCEPT

The file system can be identified as the part, so to speak, most visible of the Operating System (SILBERSCHATZ; GALVIN; GAGNE, 2004). That is, usually when a user is in front of a computer, he is presented with one or more files, or the user does this search, aiming to access a file or more files (TANENBAUM; BOS, 2016). Thus, it is notorious the importance of the file system, since it will provide interface, supports, subsidies to access the files. But, what is an archive? “A file is a collection of related information that is named and saved in secondary storage” (SILBERSCHATZ; GALVIN; GAGNE, 2004, p. 244).

According to Tanenbaum and Bos (2016):

> Files are managed by the operating system. How they are structured, named, accessed, used, protected, implemented, and managed are important topics in the design of an operating system. As a whole, that part of the operating system dealing with files is known as the file system (TANENBAUM; BOS, 2016, p. 182).

Thus, we can say that the file system is the Operating System module that is related to the persistent storage (that is, the information generated by a process must continue after the end of this process) of information on the storage disk, that is, the operating system manages the files through the file system, which consists of logical structures and routines that enable the operating system to manage access to the hard disk. It is important to note, then, that different operating systems can have different file systems, with various peculiarities. Therefore, there must be compatibility between the operating system and the file system.

FUNCTIONS / MAIN ROLES THAT A FILE SYSTEM PLAYS

In the previous section it is possible to see, roughly, the scope of the file system and thus have the scope of its functions and/or main roles that it plays. Moreover, the file system has the function of defining how the bytes that are part of the composition of files will be stored on the disk and denotes the way of accessibility of the operating system to the data in question, with the role of managing the files, access, reading, writing, storage and general use of files by the operating system.

THE NTFS FILE SYSTEM

The NTFS (New Technology File System)

file system emerged in order to have a more efficient file system, in relation to other file systems, such as, for example, the FAT 16 and FAT 32, due to the evolution of the operational systems, as, in the case of Microsoft, from Windows NT (New Technology), referring to the Windows operational systems, as well as with Windows 2000 onwards, using the NTFS, considered one of the most important existing file systems.

As an example of distinction with another file system, unlike MS - DOS, NTFS has an effective differentiation of lowercase and uppercase letters. So, when it comes to filenames, with NTFS, joseph is not the same as Joseph. Also, NTFS uses 64-bit disk addresses, and can support theoretically 264 bytes of disk partitions, which is much larger than FAT-16 and FAT-32.

STRUCTURE, IMPLEMENTATION AND DIRECTORY IMPLEMENTATION

In the NFTS file system, a file is not just a linear sequence of bytes, as in FAT - 32 file systems, for example. In NTFS, files are considered to be several attributes, each typified by a stream of bytes. Another point to note is that the NTFS file system is hierarchical.

All volumes in the NTFS file system have files, directories, bitmaps, and other data structures. According to Tanenbaum and Bos (2016), "each volume is organized as a linear sequence of blocks ('clusters' in Microsoft terminology), with block size determined for each volume and ranging from 512 bytes to 64 KB, depending on the volume size" (TANENBAUM; BOS, 2016, p. 663).

Also according to these authors, most NTFS disks use 4 KB blocks, serving as a balance point between blocks considered large, with regard to efficient transfers and blocks considered small, in order to obtain a reduced internal fragmentation (TANENBAUM; BOS, 2016).

The most important data structure of each volume is the so-called Master File Table (MFT), which consists of a linear sequence and records with a fixed size of 1 KB, where each file is described by only one MFT record. It must be noted that the MFT is also a file, where the file can grow as needed, up to the limit of 248 records (TANENBAUM; BOS, 2016).

Take a look at the MFT shown in the following figure.

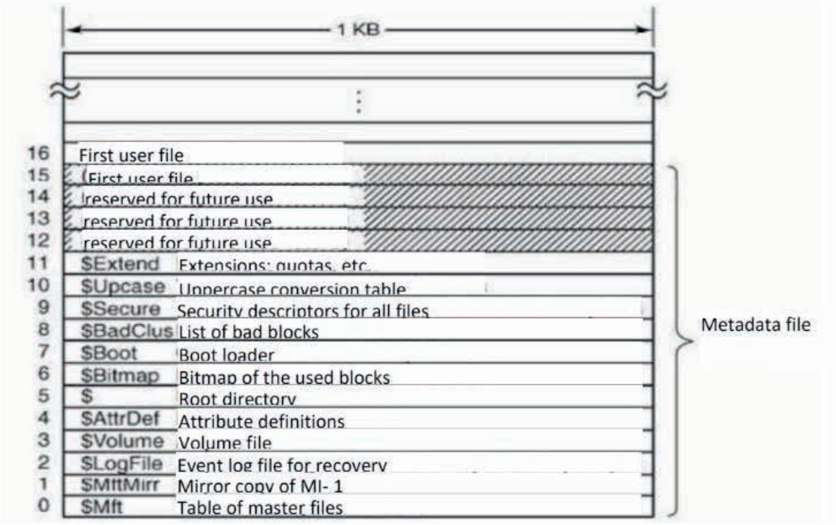
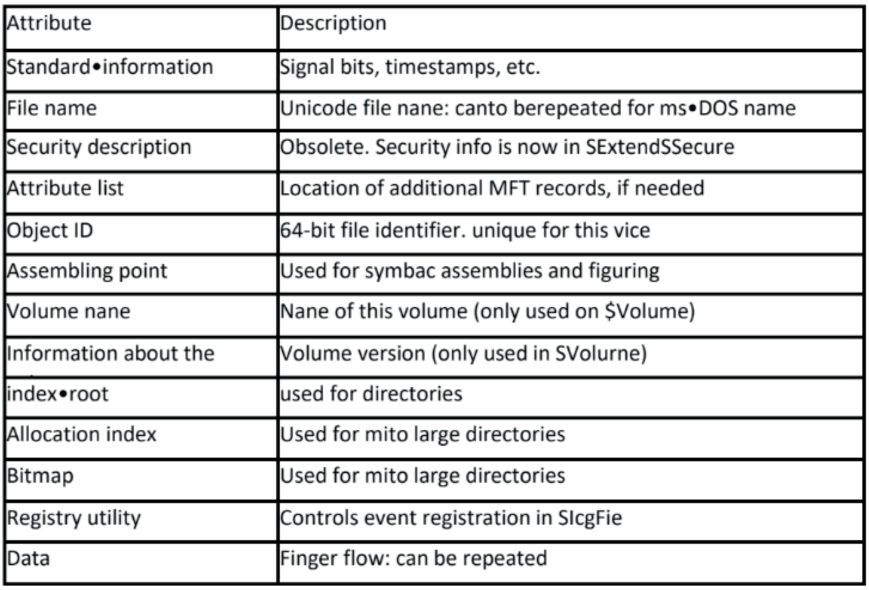

1 KB
16 First user file
15 (First user file
14 reserved for future use
13 reserved for future use
12 reserved for future use
11 $Extend Extensions: quotas, etc.
10 $Upcase Uppercase conversion table
9 $Secure Security descriptors for all files
8 $BadClus List of bad blocks
7 $Boot Boot loader
6 $Bitmap Bitmap of the used blocks
5 $ Root directory
4 $AttrDef Attribute definitions
3 $Volume Volume file
2 $LogFile Event log file for recovery
1 $MftMirr Mirror copy of MI- 1
0 $Mft Table of master files
Metadata file

Figure 1: NTFS file table

Source: (TANENBAUM; BOS, 2016, p. 663).

Notice that each record of the NTFS Master File Table composes a sequence of pairs: attribute header and value, indicating what the attribute is and the size of the value.

Note in the figure below the attributes used in the MFT records.



Attribute	Description
Standard•information	Signal bits, timestamps, etc.
File name	Unicode file nane: canto berepeated for ms•DOS name
Security description	Obsolete. Security info is now in SExtendSSecure
Attribute list	Location of additional MFT records, if needed
Object ID	64-bit file identifier. unique for this vice
Assembling point	Used for symbac assemblies and figuring
Volume nane	Nane of this volume (only used on $Volume)
Information about the	Volume version (only used in SVolurne)
index•root	used for directories
Allocation index	Used for mito large directories
Bitmap	Used for mito large directories
Registry utility	Controls event registration in SIcgFie
Data	Finger flow: can be repeated

Figure 2 - The attributes used in the MFT records.

Source: (TANENBAUM; BOS, 2016, p. 665).

The thirteen (13) attributes defined in NTFS can appear in the MFT records, as visualized in the table above. When an attribute is inserted into a disk block separately, it is known as a non-resident attribute, as may be the case with the data attribute.

Non-resident attributes are larger than resident attributes that are 24 bytes long, since non-resident attributes "contain information about where to find the attribute on the disk" (TANENBAUM; BOS, 2016, p. 664). It must be noted that the file name, in NTFS, is a field in Unicode and of a size that can vary.

METHOD OF ALLOCATION, FREE SPACE MANAGEMENT, EFFICIENCY AND PERFORMANCE, AND RETRIEVAL AND INFORMATION STORED IN THE FCB

For the sake of efficiency, the tracking of disk blocks is done as far as possible by serial assignment of consecutive blocks. The FCB (File Control Block) refers to a file control block, which is a structure containing the file's metadata and the location of its contents on disk. In the case of NTFS, file control blocks are defined in separate structures such as the MFT (Master File Table), as presented in the previous item (TANENBAUM; BOS, 2016).

Observe the following figures.

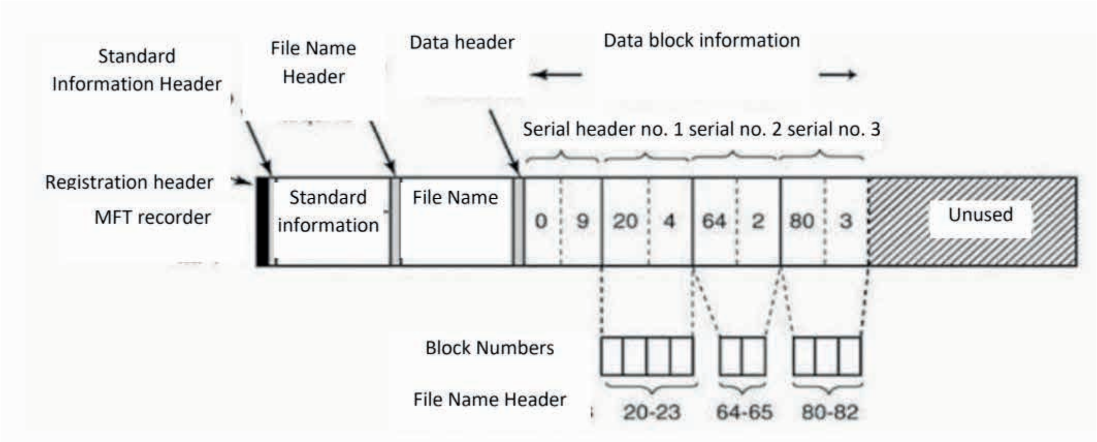


Figure 3 - An MFT record for a three-series, nine-block file.

Source: (TANENBAUM; BOS, 2016, p. 666).

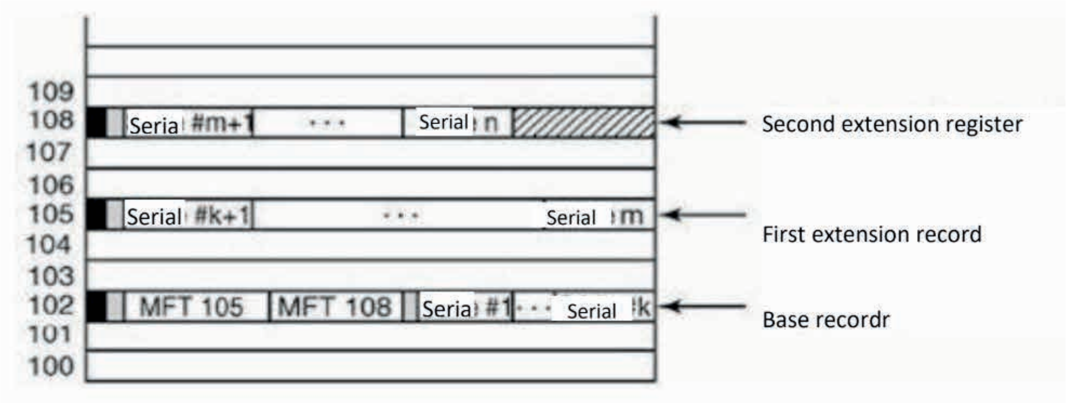


Figure 4 - A register that requires three MFT registers to store all its series.

Source: (TANENBAUM; BOS, 2016, p. 667).

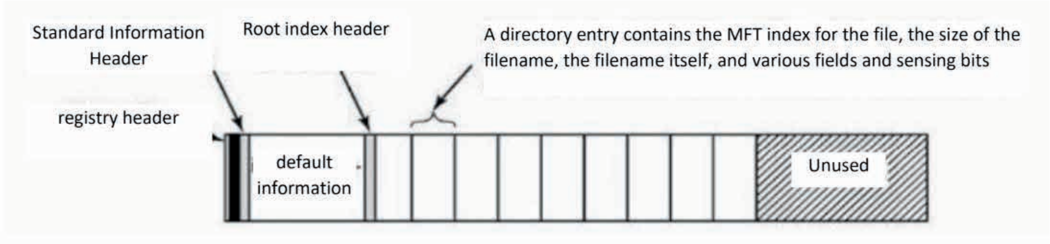

Standard Information Header
Root index header
A directory entry contains the MFT index for the file, the size of the filename, the filename itself, and various fields and sensing bits
registry header
default information
Unused

Figure 5 - The MFT registry for a small directory.

Source: (TANENBAUM; BOS, 2016, p. 667).

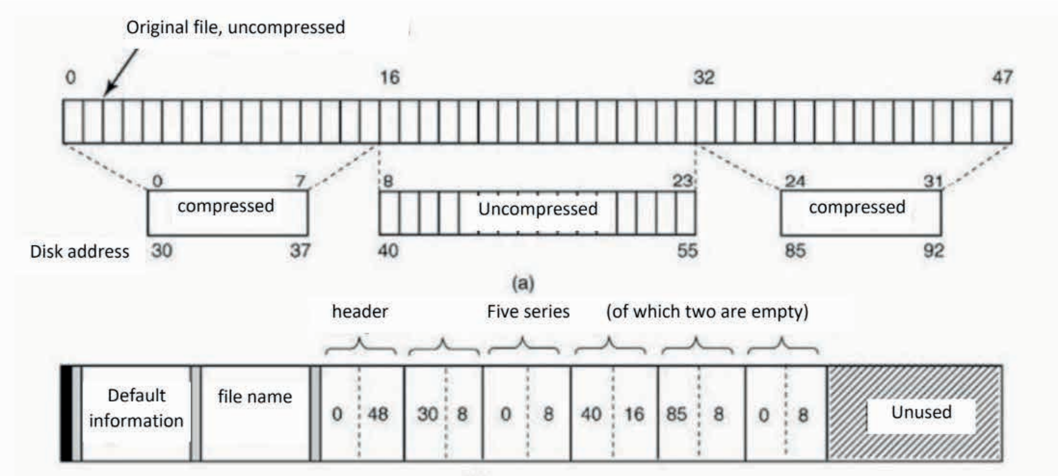


Figure 6 - (a) An example record with 48 blocks being compressed to 32 blocks. (b) The MFT record for the file after compression.

Fonte: (TANENBAUM; BOS, 2016, p. 669).

FINAL CONSIDERATIONS

According to Microsoft (2022), NTFS uses its file information and log checkpoint to restore file system consistency when the computer is restarted after a system crash. After a bad sector error, NTFS dynamically remaps the cluster containing the bad sector, allocates a new cluster for the data, marks in the original cluster as bad, and does not use the old cluster. For example, after a server crash, NTFS can recover data by replaying its log files (MICROSOFT, 2022).

REFERÊNCIAS

BERNAL, Volnys Borges. **Sistema de Arquivos**. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/3477103/mod_resource/content/1/202-SistemaArquivos.pdf>. Acesso em 07/04/2022.

MICROSOFT. **Visão Geral do NFTS**. Disponível em: <https://docs.microsoft.com/pt-br/windows-server/storage/file-server/ntfs-overview>. Acesso em 07/04/2022.

MONQUEIRO, Júlio César Bessa. NTFS: um sistema de arquivos com integridade e complexidade. Disponível em: <https://www.hardware.com.br/artigos/ntfs/>. Acesso em 07/04/2022.

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Sistemas Operacionais: conceitos e aplicações**. Rio de Janeiro: Elsevier, 2004.

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas operacionais modernos**. 4.ed. – São Paulo: Pearson Education do Brasil, 2016.