NTFS

NTFS (**New Technology File System**)[1] is a proprietary journaling file system developed by Microsoft.[1] Starting with Windows NT 3.1, it is the default file system of the Windows NT family.[9]

NTFS has several technical improvements over the file systems that it superseded – File Allocation Table (FAT) and High Performance File System (HPFS) – such as improved support for metadata and advanced data structures to improve performance, reliability, and disk space use. Additional extensions are a more elaborate security system based on access control lists (ACLs) and file system journaling.

NTFS is supported in other desktop and server operating systems as well. Linux and BSD have a free and open-source NTFS driver, called NTFS-3G, with both read and write functionality. macOS comes with read-only support for NTFS.

Contents

- **History**
- **Versions**
- **Features**
  - Scalability
  - Journaling
  - Hard links
  - Alternate data stream (ADS)
  - File compression
    - CompactOS algorithms
  - Sparse files
  - Volume Shadow Copy
  - Transactions
  - Security
  - Encryption
  - Quotas
  - Reparse points
  - Resizing
- **Internals**
  - Partition Boot Sector (VBR)
  - Master File Table
  - Metafiles
  - Attribute lists, attributes, and streams
  - Resident vs. non-resident attributes
  - Opportunistic locks

NTFS

Developer(s)	Microsoft
Full name	NT File System[1]
Introduced	July 1993 with Windows NT 3.1
Partition identifier	`0x07` (MBR) `EBD0A0A2-B9E5-4433-87C0-68B6B72699C7` (GPT)
Structures	
Directory contents	B-tree variant[2][3]
File allocation	Bitmap
Bad blocks	$BadClus (MFT Record)
Limits	
Max. volume size	2^{64} clusters − 1 cluster (format); 256 TiB − 64 KB (Windows 10 version 1703, Windows Server 2016 or earlier implementation)[4] 8 PB – 2 MB (Windows 10 version 1709, Windows Server 2019 or later implementation)[5]
Max. file size	16 EiB – 1 KB (format); 16 TB – 64 KB (Windows 7, Windows Server 2008 R2 or earlier implementation)[4]

Time
Interoperability
Implementations
See also
References
Further reading

History

In the mid-1980s, Microsoft and IBM formed a joint project to create the next generation of graphical operating system; the result was OS/2 and HPFS. Because Microsoft disagreed with IBM on many important issues, they eventually separated; OS/2 remained an IBM project and Microsoft worked to develop Windows NT and NTFS.

The HPFS file system for OS/2 contained several important new features. When Microsoft created their new operating system, they "borrowed" many of these concepts for NTFS.[10] The original NTFS developers were Tom Miller, Gary Kimura, Brian Andrew, and David Goebel.[11]

Probably as a result of this common ancestry, HPFS and NTFS use the same disk partition identification type code (07). Using the same Partition ID Record Number is highly unusual, since there were dozens of unused code numbers available, and other major file systems have their own codes. For example, FAT has more than nine (one each for FAT12, FAT16, FAT32, etc.). Algorithms identifying the file system in a partition type 07 must perform additional checks to distinguish between HPFS and NTFS.

Versions

Microsoft has released five versions of NTFS:

	256 TB – 64 KB (Windows 8, Windows Server 2012 or later implementation)[6] 8 PB – 2 MiB (Windows 10 version 1709, Windows Server 2019 or later implementation)[5]
Max. number of files	4,294,967,295 (2^{32}-1)[4]
Max. filename length	255 UTF-16 code units[7]
Allowed characters in filenames	In Win32 namespace: any UTF-16 code unit (case-insensitive) except `/\:*"?<>|` as well as NUL[7] In POSIX namespace: any UTF-16 code unit (case-sensitive) except `/` as well as NUL
Features	
Dates recorded	Creation, modification, POSIX change, access
Date range	1 January 1601 – 28 May 60056 (File times are 64-bit numbers counting 100-nanosecond intervals (ten million per second) since 1601, which is 58,000+ years)
Date resolution	100 ns
Forks	Yes (see § Alternate data

NTFS version number	First operating system	Release date	New features	Remarks
1.0	Windows NT 3.1	1993[9]	Initial version	NTFS 1.0 is incompatible with 1.1 and newer: volumes written by Windows NT 3.5x cannot be read by Windows NT 3.1 until an update (available on the NT 3.5x installation media) is installed.[12]
1.1	Windows NT 3.51	1995	Compressed files, named streams, and access control lists[13]	
1.2	Windows NT 4.0	1996	Security descriptors	Commonly called NTFS 4.0 after the OS release
3.0	Windows 2000	2000	Disk quotas, Encrypting File System, sparse files, reparse points, update sequence number (USN) journaling, the `$Extend` folder and its files	Compatibility was also made available for Windows NT 4.0 with the Service Pack 4 update. Commonly called NTFS 5.0 after the OS release.[14]
3.1	Windows XP	October 2001	Expanded the Master File Table (MFT) entries with redundant MFT record number (useful for recovering damaged MFT files)	Commonly called NTFS 5.1 after the OS release

The `NTFS.sys` version number (e.g. v5.0 in Windows 2000) is based on the operating system version; it should not be confused with the NTFS version number (v3.1 since Windows XP).[15]

Although subsequent versions of Windows added new file system-related features, they did not change NTFS itself. For example, Windows Vista implemented NTFS symbolic links, Transactional NTFS, partition shrinking, and self-healing.[16] NTFS symbolic links are a new feature in the file system; all the others are new operating system features that make use of NTFS features already in place.

	streams (ADS) below)
Attributes	Read-only, hidden, system, archive, not content indexed, off-line, temporary, compressed
File system permissions	ACLs
Transparent compression	Per-file, LZ77 (Windows NT 3.51 onward)
Transparent encryption	Per-file, DESX (Windows 2000 onward), Triple DES (Windows XP onward), AES (Windows XP Service Pack 1, Windows Server 2003 onward)
Data deduplication	Yes (Windows Server 2012)[8]
Other	
Supported operating systems	Windows NT 3.1 and later Mac OS X 10.3 and later (read-only) Linux kernel version 2.6 and later Linux kernel versions 2.2-2.4 (read-only) FreeBSD NetBSD Chrome OS Solaris ReactOS (read-only)

Features

NTFS v3.0 includes several new features over its predecessors: sparse file support, disk use quotas, reparse points, distributed link tracking, and file-level encryption called the Encrypting File System (EFS).

Scalability

NTFS is optimized for 4 KB clusters, but supports a maximum cluster size of 2 MB. (Earlier implementations support up to 64 KB)[5] The maximum NTFS volume size that the specification can support is $2^{64} - 1$ clusters, but not all implementations achieve this theoretical maximum, as discussed below.

The maximum NTFS volume size implemented in Windows XP Professional is $2^{32} - 1$ clusters, partly due to partition table limitations. For example, using 64 KB clusters, the maximum size Windows XP NTFS volume is 256 TB minus 64 KB. Using the default cluster size of 4 KB, the maximum NTFS volume size is 16 TB minus 4 KB. Both of these are vastly higher than the 128 GB limit in Windows XP SP1. Because partition tables on master boot record (MBR) disks support only partition sizes up to 2 TB, multiple GUID Partition Table (GPT or "dynamic") volumes must be combined to create a single NTFS volume larger than 2 TB. Booting from a GPT volume to a Windows environment in a Microsoft supported way requires a system with Unified Extensible Firmware Interface (UEFI) and 64-bit support.[17]

The NTFS maximum theoretical limit on the size of individual files is 16 EB[18] (16×1024^6 or 2^{64} bytes) minus 1 KB, which totals 18,446,744,073,709,550,592 bytes. With Windows 10 version 1709 and Windows Server 2019, the maximum *implemented* file size is 8 PB minus 2 MB or 9,007,199,252,643,840 bytes.[5]

Journaling

NTFS is a journaling file system and uses the NTFS Log ($LogFile) to record metadata changes to the volume. It is a feature that FAT does not provide and critical for NTFS to ensure that its complex internal data structures will remain consistent in case of system crashes or data moves performed by the defragmentation API, and allow easy rollback of uncommitted changes to these critical data structures when the volume is remounted. Notably affected structures are the volume allocation bitmap, modifications to MFT records such as moves of some variable-length attributes stored in MFT records and attribute lists, and indices for directories and security descriptors.

The ($LogFile) format has evolved through several versions:

Windows Version	$LogFile format version
Windows NT 4.0	1.1
Windows 2000	
Windows XP	
Windows Vista	
Windows 7	
Windows 8	
Windows 8.1	2.0
Windows 10	

The incompatibility of the $LogFile versions implemented by Windows 8.1 and Windows 10 prevents Windows 8 (and earlier versions of Windows) from correctly processing the $LogFile in case the NTFS volume is left in the **dirty state** by an abrupt shutdown or by hibernating to disk in the logoff state (a.k.a.: Hybrid Boot or Fast Boot, which is enabled by default in Windows 10). This inability to process the v2.0 of the $LogFile on dirty volumes by these earlier versions of Windows results in invocation of the CHKDSK disk repair utility when dual-booting Windows 10 with these older systems. A Windows Registry setting exists to prevent the automatic upgrade of the $LogFile to the newer version.[19]

The USN Journal (Update Sequence Number Journal) is a system management feature that records (in $Extend\$UsnJrnl) changes to files, streams and directories on the volume, as well as their various attributes and security settings. The journal is made available for applications to track changes to the volume.[20] This journal can be enabled or disabled on non-system volumes.[21]

Hard links

The hard link feature allows different file names to directly refer to the same file contents. Hard links may link only to files in the same volume, because each volume has its own MFT. Hard links were originally included to support the POSIX subsystem in Windows NT.[22]

Although Hard links use the same MFT record (inode) which records file metadata such as file size, modification date, and attributes, NTFS also caches this data in the directory entry as a performance enhancement. This means that when listing the contents of a directory using FindFirstFile/FindNextFile family of APIs, (equivalent to the POSIX opendir/readdir APIs) you will also receive this cached information, in addition to the name and inode. However, you may not see up-to-date information, as this information is only guaranteed to be updated when a file is closed, and then only for the directory from which the file was opened.[23] This means where a file has multiple names via hard links, updating a file via one name does not update the cached data associated with the other name. You can always obtain up-to-date data using GetFileInformationByHandle (which is the true equivalent of POSIX stat function). This can be done using a handle which has no access to the file itself (passing zero to CreateFile for dwDesiredAccess), and closing this handle has the incidental effect of updating the cached information.

Windows uses hard links to support short (8.3) filenames in NTFS. Operating system support is needed because there are legacy applications that can work only with 8.3 filenames, but support can be disabled. In this case, an additional filename record and directory entry is added, but both 8.3 and long file name are linked and updated together, unlike a regular hard link.

The NTFS file system has a limit of 1024 hard links on a file.[24]

Alternate data stream (ADS)

Alternate data streams allow more than one data stream to be associated with a filename (a fork), using the format "filename:streamname" (e.g., "text.txt:extrastream").

NTFS Streams were introduced in Windows NT 3.1, to enable Services for Macintosh (SFM) to store resource forks. Although current versions of Windows Server no longer include SFM, third-party Apple Filing Protocol (AFP) products (such as GroupLogic's ExtremeZ-IP) still use this feature of the file system. Very small ADSs (named "Zone.Identifier") are added by Internet Explorer and recently by other browsers to mark files downloaded from external sites as possibly unsafe to run; the local shell would then require user confirmation before opening them.[25] When the user indicates that he no longer wants this confirmation dialog, this ADS is deleted.

Alternate streams are not listed in Windows Explorer, and their size is not included in the file's size. When the file is copied or moved to another file system without ADS support the user is warned that alternate data streams cannot be preserved. No such warning is typically provided if the file is attached to an e-mail, or uploaded to a website. Thus, using alternate streams for critical data may cause problems. Microsoft provides a tool called Streams[26] to view streams on a selected volume. Starting with Windows PowerShell 3.0, it is possible to manage ADS natively with six cmdlets: Add-Content, Clear-Content, Get-Content, Get-Item, Remove-Item, Set-Content.[27]

Malware has used alternate data streams to hide code.[28] As a result, malware scanners and other special tools now check for alternate data streams.

File compression

NTFS can compress files using LZNT1 algorithm (a variant of LZ77)[29] Files are compressed in 16 cluster chunks. With 4 KB clusters, files are compressed in 64 KB chunks. The compression algorithms in NTFS are designed to support cluster sizes of up to 4 KB. When the cluster size is greater than 4 KB on an NTFS volume, NTFS compression is not available.[30] If the compression reduces 64 KB of data to 60 KB or less, NTFS treats the unneeded 4 KB pages like empty sparse file clusters—they are not written. This allows for reasonable random-access times as the OS merely has to follow the chain of fragments.

> ***Note:*** *The following section refers to tests, research and recommendations done and intended for storage devices with a high access time, such as a mechanical HDD, where the internal heads used for reading data, needs to be physically moved and positioned correctly, and then wait for the data on the rotating disks to pass beneath them (https://ed.ted.com/lessons/how-do-hard-drives-work-kanawat-senanan#watch). See further down for updated info regarding SSD and similar devices with low access time.*

However, large compressible files become highly fragmented since every chunk smaller than 64 KB becomes a fragment.[31][32] According to research by Microsoft's NTFS Development team, 50–60 GB is a reasonable maximum size for a compressed file on an NTFS volume with a 4 KB (default) cluster (block) size. This reasonable maximum size decreases sharply for volumes with smaller cluster sizes.[31] Single-user systems with limited hard disk space can benefit from NTFS compression for small files, from 4 KB to 64 KB or more, depending on compressibility. Files smaller than approximately 900 bytes are stored within the directory entry of the MFT.[33]

Flash memory, such as SSD drives do not have the head movement delays of hard disk drives, so fragmentation has only a smaller penalty. Users of fast multi-core processors will find improvements in application speed by compressing their applications and data as well as a reduction in space used. Note that SSDs with Sandforce controllers already compress data. However, since less data is transferred, there is a reduction in I/Os.[34]

Compression works best with files that have repetitive content, are seldom written, are usually accessed sequentially, and are not themselves compressed. Log files are an ideal example.

If system files that are needed at boot time (such as drivers, NTLDR, winload.exe, or BOOTMGR) are compressed, the system may fail to boot correctly, because decompression filters are not yet loaded.[35] Later editions of Windows do not allow important system files to be compressed.

Files may be compressed or decompressed individually (via changing the advanced attributes) for a drive, directory, or directory tree, becoming a default for the files inside.

Although read–write access to compressed files is transparent,[36] Microsoft recommends avoiding compression on servers or network shares holding roaming profiles, because it puts a considerable load on the processor.[37]

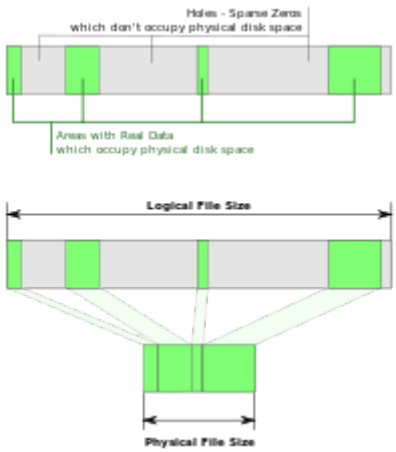
CompactOS algorithms

Since Windows 10, Microsoft has introduced additional algorithms, namely XPRESS4K/8K/16K and LZX. Both algorithms are based on LZ77 with Huffman entropy coding, which LZNT1 lacked. These algorithms were taken from the Windows Imaging Format. They are mainly used for new CompactOS feature, which compresses the entire system partition using one of these algorithms.[38] They can also be manually turned on per file with the `/exe` flag of the `compact` command. When used on files, CompactOS algorithm avoids fragmentation by writing compressed data in contiguously allocated chunks.

Sparse files

Sparse files are files interspersed with empty segments for which no actual storage space is used. To the applications, the file looks like an ordinary file with empty regions seen as regions filled with zeros.[39] A sparse file does not necessarily include sparse zeros areas; the "sparse file" attribute just means that the file is allowed to have them.

Database applications, for instance, may use sparse files.[40] As with compressed files, the actual sizes of sparse files are not taken into account when determining quota limits.[41]



A sparse file: Empty bytes don't need to be saved, thus they can be represented by metadata.

Volume Shadow Copy

The Volume Shadow Copy Service (VSS) keeps historical versions of files and folders on NTFS volumes by copying old, newly overwritten data to shadow copy via copy-on-write technique. The user may later request an earlier version to be recovered. This also allows data backup programs to archive files currently in use by the file system. On heavily loaded systems, Microsoft recommends setting up a shadow copy volume on a separate disk.[42]

Windows Vista also introduced persistent shadow copies for use with System Restore and Previous Versions features. Persistent shadow copies, however, are deleted when an older operating system mounts that NTFS volume. This happens because the older operating system does not understand the newer format of persistent shadow copies.[43]

Transactions

As of Windows Vista, applications can use Transactional NTFS (TxF) to group multiple changes to files together into a single transaction. The transaction will guarantee that either all of the changes happen, or none of them do, and that no application outside the transaction will see the changes until they are committed.[44]

It uses similar techniques as those used for Volume Shadow Copies (i.e. copy-on-write) to ensure that overwritten data can be safely rolled back, and a CLFS log to mark the transactions that have still not been committed, or those that have been committed but still not fully applied (in case of system crash during a commit by one of the participants).

Transactional NTFS does not restrict transactions to just the local NTFS volume, but also includes other transactional data or operations in other locations such as data stored in separate volumes, the local registry, or SQL databases, or the current states of system services or remote services. These transactions are coordinated network-wide with all participants using a specific service, the DTC, to ensure that all participants will receive same commit state, and to transport the changes that have been validated by any participant (so that the others can invalidate their local caches for old data or rollback their ongoing uncommitted changes). Transactional NTFS allows, for example, the creation of network-wide consistent distributed file systems, including with their local live or offline caches.

Microsoft now advises against using TxF: "Microsoft strongly recommends developers utilize alternative means" since "TxF may not be available in future versions of Microsoft Windows".[45]

Security

In NTFS, each file or folder is assigned a security descriptor that defines its owner and contains two access control lists (ACLs). The first ACL, called discretionary access control list (DACL), defines exactly what type of interactions (e.g. reading, writing, executing or deleting) are allowed or forbidden by which user or groups of users. For example, files in the `C:\Program Files` folder may be read and executed by all users but modified only by a user holding administrative privileges.[46] Windows Vista adds mandatory access control info to DACLs. DACLs are the primary focus of User Account Control in Windows Vista and later.

The second ACL, called system access control list (SACL), defines which interactions with the file or folder are to be audited and whether they should be logged when the activity is successful, failed or both. For example, auditing can be enabled on sensitive files of a company, so that its managers get to know when someone tries to delete them or make a copy of them, and whether he or she succeeds.[46]

Encryption

Encrypting File System (EFS) provides strong[47] and user-transparent encryption of any file or folder on an NTFS volume. EFS works in conjunction with the EFS service, Microsoft's CryptoAPI and the EFS File System Run-Time Library (FSRTL). EFS works by encrypting a file with a bulk symmetric key (also known as the File Encryption Key, or FEK), which is used because it takes a relatively small amount of time to encrypt and decrypt large amounts of data than if an asymmetric key cipher is used. The symmetric key that is used to encrypt the file is then encrypted with a public key that is associated with the user who encrypted the file, and this encrypted data is stored in an alternate data stream of the encrypted file. To decrypt the file, the file system uses the private key of the user to decrypt the symmetric key that is stored in the data stream. It then uses the symmetric key to decrypt the file. Because this is done at the file system level, it is transparent to the user.[48] Also, in case of a user losing access to their key, support for additional decryption keys has been built into the EFS system, so that a recovery agent can still access the files if needed. NTFS-provided encryption and NTFS-provided compression are mutually exclusive; however, NTFS can be used for one and a third-party tool for the other.

The support of EFS is not available in Basic, Home, and MediaCenter versions of Windows, and must be activated after installation of Professional, Ultimate, and Server versions of Windows or by using enterprise deployment tools within Windows domains.

Quotas

Disk quotas were introduced in NTFS v3. They allow the administrator of a computer that runs a version of Windows that supports NTFS to set a threshold of disk space that users may use. It also allows administrators to keep track of how much disk space each user is using. An administrator may specify a certain level of disk space that a user may use before they receive a warning, and then deny access to the user once they hit their upper limit of space. Disk quotas do not take into account NTFS's transparent file-compression, should this be enabled. Applications that query the amount of free space will also see the amount of free space left to the user who has a quota applied to them.

Reparse points

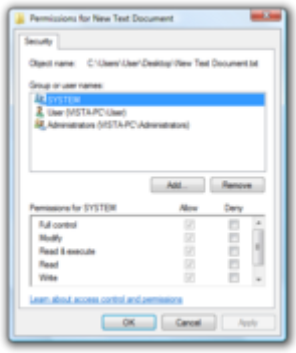
Introduced in NTFS v3, NTFS reparse points are used by associating a reparse tag in the user space attribute of a file or directory. Microsoft includes several default tags including symbolic links, directory junction points and volume mount points. When the Object Manager parses a file system name lookup and encounters a reparse attribute, it will *reparse* the name lookup, passing the user controlled reparse data to every file system filter driver that is loaded into Windows. Each filter driver examines the reparse data to see whether it is associated with that reparse point, and if that filter driver determines a match, then it intercepts the file system request and performs its special functionality.

Resizing

Starting with Windows Vista Microsoft added the built-in ability to shrink or expand a partition. However, this ability does not relocate page file fragments or files that have been marked as unmovable, so shrinking a volume will often require relocating or disabling any page file, the index of Windows Search, and any Shadow Copy used by System Restore. Various third-party tools are capable of resizing NTFS partitions.

Internals

Internally, NTFS uses B-trees to index file system data. A file system journal is used to guarantee the integrity of the file system metadata but not individual files' content. Systems using NTFS are known to have improved reliability compared to FAT file systems.[49]



NTFS file system permissions on a Windows Vista system

NTFS allows any sequence of 16-bit values for name encoding (file names, stream names, index names, etc.) except 0x0000. This means UTF-16 code units are supported, but the file system does not check whether a sequence is valid UTF-16 (it allows any sequence of short values, not restricted to those in the Unicode standard). In Win32 namespace, any UTF-16 code units are case insensitive whereas in POSIX namespace they are case sensitive. File names are limited to 255 UTF-16 code units. Certain names are reserved in the volume root directory and cannot be used for files. These are `$MFT`, `$MFTMirr`, `$LogFile`, `$Volume`, `$AttrDef`, `.` (dot), `$Bitmap`, `$Boot`, `$BadClus`, `$Secure`, `$UpCase`, and `$Extend`.[4] `.` (dot) and `$Extend` are both directories; the others are files. The NT kernel limits full paths to 32,767 UTF-16 code units. There are some additional restrictions on code points and file names.[50]

Partition Boot Sector (VBR)

NTFS boot sector contents[51][52] (All values except strings are stored in little endian order.)

Byte offset	Field length	Typical value	Field name		Purpose
0x00	3 bytes	0xEB5290	JMP instruction		Causes execution to continue after the data structures in this boot sector.
0x03	8 bytes	"NTFS " Word "NTFS" followed by four trailing spaces (0x20)	OEM ID		This is the magic number that indicates this is an NTFS file system.
0x0B	2 bytes	0x0200	BPB	Bytes per sector	The number of bytes in a disk sector.
0x0D	1 byte	0x08		Sectors Per Cluster	The number of sectors in a cluster. If the value is greater than 0x80, the amount of sectors is 2 to the power of the absolute value of considering this field to be negative.
0x0E	2 bytes	0x0000		Reserved Sectors, unused	How much space is reserved by the OS at the start of disk. This is always 9.
0x10	3 bytes	0x000000		Unused	This field is always 0
0x13	2 bytes	0x0000		Unused by NTFS	This field is always 0
0x15	1 byte	0xF8		Media Descriptor	The type of drive. 0xF8 is used to denote a hard drive (in contrast to the several sizes of floppy).
0x16	2 bytes	0x0000		Unused	This field is always 0
0x18	2 bytes	0x003F		Sectors Per Track	The number of disk sectors in a drive track.
0x1A	2 bytes	0x00FF		Number Of Heads	The number of heads on the drive.
0x1C	4 bytes	0x0000003F		Hidden Sectors	The number of sectors preceding the partition.
0x20	4 bytes	0x00000000		Unused	Not used by NTFS
0x24	4 bytes	0x00800080	EBPB	Unused	Not used by NTFS
0x28	8 bytes	0x00000000007FF54A		Total sectors	The partition size in sectors.
0x30	8 bytes	0x0000000000000004		$MFT cluster number	The cluster that contains the Master File Table
0x38	8 bytes	0x000000000007FF54		$MFTMirr cluster number	The cluster that contains a backup of the Master File Table
0x40	1 byte	0xF6		Bytes or Clusters Per File Record Segment	A positive value denotes the number of clusters in a File Record Segment. A negative value denotes the amount of bytes in a File Record Segment, in which case the size is 2 to the power of the absolute value. (0xF6 = -10 → 2^{10} = 1024).
0x41	3	0x000000		Unused	This field is not used by NTFS

	bytes				
0x44	1 byte	0x01		Bytes or Clusters Per Index Buffer	A positive value denotes the number of clusters in an Index Buffer. A negative value denotes the amount of bytes and it uses the same algorithm for negative numbers as the "Bytes or Clusters Per File Record Segment."
0x45	3 bytes	0x000000		Unused	This field is not used by NTFS
0x48	8 bytes	0x1C741BC9741BA514		Volume Serial Number	A unique random number assigned to this partition, to keep things organized.
0x50	4 bytes	0x00000000		Checksum, unused	Supposedly a checksum.
0x54	426 bytes		Bootstrap Code		The code that loads the rest of the operating system. This is pointed to by the first 3 bytes of this sector.
0x01FE	2 bytes	0xAA55	End-of-sector Marker		This flag indicates that this is a valid boot sector.

This boot partition format is roughly based upon the earlier FAT filesystem, but the fields are in different locations. Some of these fields, especially the "sectors per track", "number of heads" and "hidden sectors" fields may contain dummy values on drives where they either do not make sense or are not determinable.

The OS first looks at the 8 bytes at 0x30 to find the cluster number of the $MFT, then multiplies that number by the number of sectors per cluster (1 byte found at 0x0D). This value is the sector offset (LBA) to the $MFT, which is described below.

Master File Table

In NTFS, all file, directory and metafile data—file name, creation date, access permissions (by the use of access control lists), and size—are stored as metadata in the **Master File Table** (**MFT**). This abstract approach allowed easy addition of file system features during Windows NT's development—an example is the addition of fields for indexing used by the Active Directory software. This also enables fast file search software to locate named local files and folders included in the MFT very quickly, without requiring any other index.

The MFT structure supports algorithms which minimize disk fragmentation.[53] A directory entry consists of a filename and a "file ID" (analogous to the inode number), which is the record number representing the file in the Master File Table. The file ID also contains a reuse count to detect stale references. While this strongly resembles the W_FID of Files-11, other NTFS structures radically differ.

Two copies of the MFT are stored in case of corruption. If the first record is corrupted, NTFS reads the second record to find the MFT mirror file. Locations for both files are stored in the boot sector.[54]

Metafiles

NTFS contains several files that define and organize the file system. In all respects, most of these files are structured like any other user file ($Volume being the most peculiar), but are not of direct interest to file system clients.[55] These metafiles define files, back up critical file system data, buffer file system changes, manage free space allocation, satisfy BIOS expectations, track bad allocation units, and store security and disk space usage information. All content is in an unnamed data stream, unless otherwise indicated.

MFT (entries 0–26 are the NTFS metafiles)

Segment number	File name	Purpose
0	`$MFT`	Describes all files on the volume, including file names, timestamps, stream names, and lists of cluster numbers where data streams reside, indexes, security identifiers, and file attributes like "read only", "compressed", "encrypted", etc.
1	`$MFTMirr`	Duplicate of the first vital entries of $MFT, usually 4 entries (4 kilobytes).
2	`$LogFile`	Contains transaction log of file system metadata changes.
3	`$Volume`	Contains information about the volume, namely the volume object identifier, volume label, file system version, and volume flags (mounted, chkdsk requested, requested $LogFile resize, mounted on NT 4, volume serial number updating, structure upgrade request). This data is not stored in a data stream, but in special MFT attributes: If present, a volume object ID is stored in an $OBJECT_ID record; the volume label is stored in a $VOLUME_NAME record, and the remaining volume data is in a $VOLUME_INFORMATION record. Note: volume serial number is stored in file $Boot (below).
4	`$AttrDef`	A table of MFT attributes that associates numeric identifiers with names.
5	`.`	Root directory. Directory data is stored in $INDEX_ROOT and $INDEX_ALLOCATION attributes both named $I30.
6	`$Bitmap`	An array of bit entries: each bit indicates whether its corresponding cluster is used (allocated) or free (available for allocation).
7	`$Boot`	Volume boot record (VBR). This file is always located at the first clusters on the volume. It contains bootstrap code (see NTLDR/BOOTMGR) and a BIOS parameter block including a volume serial number and cluster numbers of $MFT and $MFTMirr.
8	`$BadClus`	A file that contains all the clusters marked as having bad sectors. This file simplifies cluster management by the chkdsk utility, both as a place to put newly discovered bad sectors, and for identifying unreferenced clusters. This file contains two data streams, even on volumes with no bad sectors: an unnamed stream contains bad sectors—it is zero length for perfect volumes; the second stream is named $Bad and contains all clusters on the volume not in the first stream.
9	`$Secure`	Access control list database that reduces overhead having many identical ACLs stored with each file, by uniquely storing these ACLs only in this database (contains two indices: $SII *(Standard_Information ID)* and $SDH *(Security Descriptor Hash)*, which index the stream named $SDS containing actual ACL table).[13]
10	`$UpCase`	A table of unicode uppercase characters for ensuring case-insensitivity in Win32 and DOS namespaces.
11	`$Extend`	A file system directory containing various optional extensions, such as $Quota, $ObjId, $Reparse or $UsnJrnl.
12–23	Reserved for $MFT extension entries. Extension entries are additional MFT records that contain additional attributes that do not fit in the primary record. This could occur if the file is sufficiently fragmented, has many streams, long filenames, complex security, or other rare situations.	
24	`$Extend\$Quota`	Holds disk quota information. Contains two index roots, named $O and $Q.
25	`$Extend\$ObjId`	Holds link tracking information. Contains an index root and allocation named $O.
26	`$Extend\$Reparse`	Holds reparse point data (such as symbolic links). Contains an index root and allocation named $R.
27–	Beginning of regular file entries.	

These metafiles are treated specially by Windows, handled directly by the `NTFS.SYS` driver and are difficult to directly view: special purpose-built tools are needed.[56] As of Windows 7, the NTFS driver completely prohibits user access, resulting in a BSoD whenever an attempt to execute a metadata file is made. One such tool is the nfi.exe ("NTFS File Sector Information Utility") that is freely distributed as part of the Microsoft "OEM Support Tools". For example, to obtain information on the "$MFT"-Master File Table Segment the following command is used: `nfi.exe c:\$MFT`[57] Another way to bypass the restriction is to use 7-Zip's file manager and go to the low-level NTFS path `\\.\X:\` (where `X:\` resembles any drive/partition). Here, 3 new folders will appear: `$EXTEND`, `[DELETED]` (a pseudo-folder that 7-Zip uses to attach files deleted from the file system to view), and `[SYSTEM]` (another pseudo-folder that contains all the NTFS metadata files). This trick can be used from removable devices (USB flash drives, external hard drives, SD Cards, etc.) inside Windows, but doing this on the active partition requires offline access (namely WinRE).

Attribute lists, attributes, and streams

For each file (or directory) described in the MFT record, there is a linear repository of stream descriptors (also named *attributes*), packed together in one or more MFT records (containing the so-called *attributes list*), with extra padding to fill the fixed 1 KB size of every MFT record, and that fully describes the effective streams associated with that file.

Each attribute has an attribute type (a fixed-size integer mapping to an attribute definition in file $AttrDef), an optional attribute name (for example, used as the name for an alternate data stream), and a value, represented in a sequence of bytes. For NTFS, the standard data of files, the alternate data streams, or the index data for directories are stored as attributes.

According to $AttrDef, some attributes can be either resident or non-resident. The $DATA attribute, which contains file data, is such an example. When the attribute is resident (which is represented by a flag), its value is stored directly in the MFT record. Otherwise, clusters are allocated for the data, and the cluster location information is stored as data runs in the attribute.

- For each file in the MFT, the attributes identified by *attribute type, attribute name* must be unique. Additionally, NTFS has some ordering constraints for these attributes.
- There is a predefined null attribute type, used to indicate the end of the list of attributes in one MFT record. It must be present as the last attribute in the record (all other storage space available after it will be ignored and just consists of padding bytes to match the record size in the MFT).
- Some attribute types are required and must be present in each MFT record, except unused records that are just indicated by null attribute types.
    - This is the case for the $STANDARD_INFORMATION attribute that is stored as a fixed-size record and contains the timestamps and other basic single-bit attributes (compatible with those managed by FAT in DOS or Windows 9x).
- Some attribute types cannot have a name and must remain anonymous.
    - This is the case for the standard attributes, or for the preferred NTFS "filename" attribute type, or the "short filename" attribute type, when it is also present (for compatibility with DOS-like applications, see below). It is also possible for a file to contain only a short filename, in which case it will be the preferred one, as listed in the Windows Explorer.
    - The filename attributes stored in the attribute list do not make the file immediately accessible through the hierarchical file system. In fact, all the filenames must be indexed separately in at least one other directory on the same volume. There it must have its own MFT record and its own security descriptors and attributes that reference the MFT record

number for this file. This allows the same file or directory to be "hardlinked" several times from several containers on the same volume, possibly with distinct filenames.

- The default data stream of a regular file is a stream of type $DATA but with an anonymous name, and the ADSs are similar but must be named.
- On the other hand, the default data stream of directories has a distinct type, but are not anonymous: they have an attribute name ("$I30" in NTFS 3+) that reflects its indexing format.

All attributes of a given file may be displayed by using the nfi.exe ("NTFS File Sector Information Utility") that is freely distributed as part of the Microsoft "OEM Support Tools".[57]

Windows system calls may handle alternate data streams.[4] Depending on the operating system, utility and remote file system, a file transfer might silently strip data streams.[4] A safe way of copying or moving files is to use the BackupRead and BackupWrite system calls, which allow programs to enumerate streams, to verify whether each stream should be written to the destination volume and to knowingly skip unwanted streams.[4]

Resident vs. non-resident attributes

To optimize the storage and reduce the I/O overhead for the very common case of attributes with very small associated value, NTFS prefers to place the value within the attribute itself (if the size of the attribute does not then exceed the maximum size of an MFT record), instead of using the MFT record space to list clusters containing the data; in that case, the attribute will not store the data directly but will just store an allocation map (in the form of *data runs*) pointing to the actual data stored elsewhere on the volume.[58] When the value can be accessed directly from within the attribute, it is called "resident data" (by computer forensics workers). The amount of data that fits is highly dependent on the file's characteristics, but 700 to 800 bytes is common in single-stream files with non-lengthy filenames and no ACLs.

- Some attributes (such as the preferred filename, the basic file attributes) cannot be made non-resident. For non-resident attributes, their allocation map must fit within MFT records.
- Encrypted-by-NTFS, sparse data streams, or compressed data streams cannot be made resident.
- The format of the allocation map for non-resident attributes depends on its capability of supporting sparse data storage. In the current implementation of NTFS, once a non-resident data stream has been marked and converted as sparse, it cannot be changed back to non-sparse data, so it cannot become resident again, unless this data is fully truncated, discarding the sparse allocation map completely.
- When a non-resident attribute is so fragmented, that its effective allocation map cannot fit entirely within one MFT record, NTFS stores the attribute in multiple records. The first one among them is called the base record, while the others are called extension records. NTFS creates a special attribute $ATTRIBUTE_LIST to store information mapping different parts of the long attribute to the MFT records, which means the allocation map may be split into multiple records. The $ATTRIBUTE_LIST itself can also be non-resident, but its own allocation map must fit within one MFT record.
- When there are too many attributes for a file (including ADS's, extended attributes, or security descriptors), so that they cannot fit all within the MFT record, extension records may also be used to store the other attributes, using the same format as the one used in the base MFT record, but without the space constraints of one MFT record.

The allocation map is stored in a form of *data runs* with compressed encoding. Each data run represents a contiguous group of clusters that store the attribute value. For files on a multi-GB volume, each entry can be encoded as 5 to 7 bytes, which means a 1 KB MFT record can store about 100 such data runs. However, as

the $ATTRIBUTE_LIST also has a size limit, it is dangerous to have more than 1 million fragments of a single file on an NTFS volume, which also implies that it is in general not a good idea to use NTFS compression on a file larger than 10 GB.[59]

The NTFS file system driver will sometimes attempt to relocate the data of some of the attributes that can be made non-resident into the clusters, and will also attempt to relocate the data stored in clusters back to the attribute inside the MFT record, based on priority and preferred ordering rules, and size constraints.

Since resident files do not directly occupy clusters ("allocation units"), it is possible for an NTFS volume to contain more files on a volume than there are clusters. For example, a 74.5 GB partition NTFS formats with 19,543,064 clusters of 4 KB. Subtracting system files (a 64 MB log file, a 2,442,888-byte Bitmap file, and about 25 clusters of fixed overhead) leaves 19,526,158 clusters free for files and indices. Since there are four MFT records per cluster, this volume theoretically could hold almost $4 \times 19{,}526{,}158 = 78{,}104{,}632$ resident files.

Opportunistic locks

Opportunistic file locks (oplocks) allow clients to alter their buffering strategy for a given file or stream in order to increase performance and reduce network use.[60] Oplocks apply to the given open stream of a file and do not affect oplocks on a different stream.

Oplocks can be used to transparently access files in the background. A network client may avoid writing information into a file on a remote server if no other process is accessing the data, or it may buffer read-ahead data if no other process is writing data.

Windows supports four different types of oplocks:

- Level 2 (or shared) oplock: multiple readers, no writers (i.e. read caching).
- Level 1 (or exclusive) oplock: exclusive access with arbitrary buffering (i.e. read and write caching).
- Batch oplock (also exclusive): a stream is opened on the server, but closed on the client machine (i.e. read, write and handle caching).
- Filter oplock (also exclusive): applications and file system filters can "back out" when others try to access the same stream (i.e. read and write caching) (since Windows 2000)

Opportunistic locks have been enhanced in Windows 7 and Windows Server 2008 R2 with per-client oplock keys.[61]

Time

Windows NT and its descendants keep internal timestamps as UTC and make the appropriate conversions for display purposes; all NTFS timestamps are in UTC.

For historical reasons, the versions of Windows that do not support NTFS all keep time internally as local zone time, and therefore so do all file systems – other than NTFS – that are supported by current versions of Windows. This means that when files are copied or moved between NTFS and non-NTFS partitions, the OS needs to convert timestamps on the fly. But if some files are moved when daylight saving time (DST) is in effect, and other files are moved when standard time is in effect, there can be some ambiguities in the conversions. As a result, especially shortly after one of the days on which local zone time changes, users may

observe that some files have timestamps that are incorrect by one hour. Due to the differences in implementation of DST in different jurisdictions, this can result in a potential timestamp error of up to 4 hours in any given 12 months.[62]

Interoperability

While the different NTFS versions are for the most part fully forward- and backward-compatible, there are technical considerations for mounting newer NTFS volumes in older versions of Microsoft Windows. This affects dual-booting, and external portable hard drives. For example, attempting to use an NTFS partition with "Previous Versions" (a.k.a. Volume Shadow Copy) on an operating system that does not support it will result in the contents of those previous versions being lost.[63] A Windows command-line utility called convert.exe can convert supporting file systems to NTFS, including HPFS (only on Windows NT 3.1, 3.5, and 3.51), FAT16 and FAT32 (on Windows 2000 and later).[64][65]

As of Windows 10 version 1709, known as the Fall Creators Update, Microsoft requires the OneDrive file structure to reside on an NTFS disk.[66] This is because OneDrive Files On-Demand feature uses NTFS reparse points to link files and folders that are stored in OneDrive to the local filesystem, thus making the file or folder unusable with any previous version of Windows, with any other NTFS file system driver, or any file system and backup utilities not updated to support it.[67]

Implementations

FreeBSD 3.2 released in May 1999 included read-only NTFS support written by Semen Ustimenko.[68][69] This implementation was ported to NetBSD by Christos Zoulas and Jaromir Dolecek and released with NetBSD 1.5 in December 2000.[70] The FreeBSD implementation of NTFS was also ported to OpenBSD by Julien Bordet and offers native read-only NTFS support by default on i386 and amd64 platforms as of version 4.9 released 1 May 2011.[71][69]

Linux kernel versions 2.2.0 and later include the ability to read NTFS partitions; kernel versions 2.6.0 and later contain a driver written by Anton Altaparmakov (University of Cambridge) and Richard Russon which supports file read, overwrite and resize. Due to the complexity of internal NTFS structures the built-in 2.6.14 kernel driver disallows changes to the volume that are considered unsafe, to avoid corruption.

Mac OS X 10.3 included Ustimenko's read-only implementation of NTFS from FreeBSD. Then Apple hired Anton Altaparmakov to write a new NTFS implementation for Mac OS X 10.6.[72] Native NTFS write support has been discovered in 10.6 and later, but is not activated by default, although workarounds do exist to enable the functionality. However, user reports indicate the functionality is unstable and tends to cause kernel panics, probably the reason why write support has not been enabled or advertised.[73]

Captive NTFS, a 'wrapping' driver that uses Windows' own driver `ntfs.sys`, exists for Linux. It was built as a Filesystem in Userspace (FUSE) program and released under the GPL but work on Captive NTFS ceased in 2006.[74]

NTFS-3G is a free GPL-licensed FUSE implementation of NTFS that was initially developed as a Linux kernel driver by Szabolcs Szakacsits. It was re-written as a FUSE program to work on other systems that FUSE supports like macOS, FreeBSD, NetBSD, OpenBSD,[75] Solaris, QNX, and Haiku[76] and allows reading and writing to NTFS partitions. A performance enhanced commercial version of NTFS-3G, called "Tuxera NTFS for Mac", is also available from the NTFS-3G developers.[77]

Paragon Software Group sells a read-write driver named *NTFS for Mac OS X*,[78] which is also included on some models of Seagate hard drives.[79]

The NetDrive package for OS/2 (and derivatives such as eComStation and ArcaOS) supports a plugin which allows read and write access to NTFS volumes.[80][81]

There is a free-for-personal-use read/write driver for MS-DOS by Avira called "NTFS4DOS".[82][83]

Ahead Software developed a "NTFSREAD" driver (version 1.200) for DR-DOS 7.0x between 2002 and 2004. It was part of their Nero Burning ROM software.

See also

- Comparison of file systems
- NTFSDOS
- ntfsresize
- WinFS (a canceled Microsoft filesystem)
- ReFS, a newer Microsoft filesystem

References

1. "Glossary" (https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/230807ac-20be-494f-86e3-4c8ac23ea584). *[MS-EFSR]: Encrypting File System Remote (EFSRPC) Protocol*. Microsoft. 14 November 2013.
2. "How NTFS Works" (https://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx). *TechNet*. Microsoft. Retrieved 2 December 2017.
3. "B*Trees - NTFS Directory Trees - Conecpt - NTFS Documentation" (https://flatcap.org/linux-ntfs/ntfs/concepts/tree/index.html). *flatcap.org*. Retrieved 2019-05-13.
4. "How NTFS Works" (https://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx). *Windows Server 2003 Technical Reference*. 2003-03-28. Retrieved 2011-09-12.
5. "Appendix A: Product Behavior" (https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-fsa/4e3695bd-7574-4f24-a223-b4679c065b63#Appendix_A_1). *[MS-FSA]: File System Algorithms*. Microsoft. 2018-09-12. Retrieved 2018-10-01. "NTFS uses a default cluster size of 4 KB, a maximum cluster size of 64 KB on Windows 10 v1703 operating system and Windows Server 2016 and prior, and 2 MB on Windows 10 v1709 operating system and Windows Server 2019 and later, and a minimum cluster size of 512 bytes."
6. "Appendix A: Product Behavior" (https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-fsa/4e3695bd-7574-4f24-a223-b4679c065b63). *[MS-FSA]: File System Algorithms*. Microsoft. 14 November 2013. Retrieved 2012-09-21.
7. Russon, Richard; Fledel, Yuval. "NTFS Documentation" (http://dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf) (PDF). Retrieved 2011-06-26.
8. Rick Vanover. "Windows Server 8 data deduplication" (https://www.techrepublic.com/blog/the-enterprise-cloud/windows-server-8-data-deduplication-what-you-need-to-know/). Retrieved 2011-12-02.
9. Custer, Helen (1994). *Inside the Windows NT File System* (https://archive.org/details/insidewindowsntf00cust). Microsoft Press. ISBN 978-1-55615-660-1.
10. Kozierok, Charles. "Overview and History of NTFS" (https://www.karlstechnology.com/blog/history-of-ntfs/). The PC Guide. Retrieved May 30, 2019.
11. Custer, Helen (1994). *Inside the Windows NT File System* (https://archive.org/details/insidewindowsntf00cust). Microsoft Press. p. vii. ISBN 978-1-55615-660-1.
12. "Recovering Windows NT After a Boot Failure on an NTFS Drive" (http://support.microsoft.com/kb/q129102/). Microsoft. November 1, 2006.

13. Russinovich, Mark. "Inside Win2K NTFS, Part 1" (https://docs.microsoft.com/en-us/previous-versions/ms995846(v=msdn.10)). *MSDN*. Microsoft. Retrieved 2008-04-18.
14. "What's New in Windows NT 4.0 Service Pack 4?" (https://web.archive.org/web/19990117055557/http://www.microsoft.com/ntserver/nts/exec/overview/NT4SP4whatnew.asp). *Microsoft.com*. 12 January 1999. Archived from the original (http://www.microsoft.com/ntserver/nts/exec/overview/NT4SP4whatnew.asp) on 17 January 1999. Retrieved 17 August 2018.
15. "New Capabilities and Features of the NTFS 3.1 File System" (http://support.microsoft.com/kb/310749). Microsoft. 1 December 2007.
16. Loveall, John (2006). "Storage improvements in Windows Vista and Windows Server 2008" (http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/STO123_WH06.ppt) (PowerPoint). Microsoft. pp. 14–20. Retrieved 2007-09-04.
17. "Booting from GPT" (http://www.rodsbooks.com/gdisk/booting.html). *Rodsbooks.com*. Retrieved 22 September 2018.
18. "NTFS vs FAT vs exFAT - NTFS.com" (https://www.ntfs.com/ntfs_vs_fat.htm). *www.ntfs.com*. Retrieved 2021-01-19.
19. https://www.prime-expert.com/articles/b26/stop-disk-check-from-running-on-every-boot/
20. "Change Journals (Windows)" (http://msdn.microsoft.com/en-us/library/aa363798.aspx). MSDN. Retrieved 2010-04-16.
21. "Creating, Modifying, and Deleting a Change Journal (Windows)" (http://msdn.microsoft.com/en-us/library/aa363877(v=VS.85).aspx). MSDN. Retrieved 2010-04-16.
22. "Chapter 29 – POSIX Compatibility" (https://technet.microsoft.com/en-us/library/cc768080.aspx). *MS Windows NT Workstation 4.0 Resource Guide*. Microsoft. 1995. Retrieved 21 October 2013.
23. "Hard Links and Junctions" (http://msdn.microsoft.com/en-us/library/aa365006%28VS.85%29.aspx). *MSDN*. Microsoft. 12 October 2013. Retrieved 21 October 2013.
24. "MSDN – CreateHardLink function" (https://msdn.microsoft.com/en-us/library/aa363860%28v=vs.85%29.aspx). Retrieved 14 January 2016.
25. Russinovich, Mark E.; Solomon, David A.; Ionescu, Alex (2009). "File Systems". *Windows Internals* (5th ed.). Microsoft Press. p. 921. ISBN 978-0-7356-2530-3. "One component in Windows that uses multiple data streams is the Attachment Execution Service[...] depending on which zone the file was downloaded from [...] Windows Explorer might warn the user"
26. "Streams - Windows Sysinternals" (https://technet.microsoft.com/en-us/sysinternals/bb897440.aspx). *Technet.microsoft.com*. Retrieved 22 September 2018.
27. "FileSystem Provider" (https://web.archive.org/web/20150123140513/https://technet.microsoft.com/en-us/library/hh847764(v=wps.620).aspx). Microsoft. 9 August 2012. Archived from the original (https://technet.microsoft.com/en-us/library/hh847764(v=wps.620).aspx) on 23 January 2015. Retrieved 23 January 2015.
28. Malware utilising Alternate Data Streams? (https://www.auscert.org.au/render.html?it=7967) Archived (https://web.archive.org/web/20080723180729/https://www.auscert.org.au/render.html?it=7967) 2008-07-23 at the Wayback Machine, AusCERT Web Log, 21 August 2007
29. "File Compression and Decompression" (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/file_compression_and_decompression.asp). MSDN Platform SDK: File Systems. Retrieved 2005-08-18.
30. "The Default Cluster Size for the NTFS and FAT File Systems" (http://support.microsoft.com/kb/314878). Microsoft. January 31, 2002. Retrieved 2012-01-10.
31. Middleton, Dennis. "Understanding NTFS Compression" (http://blogs.msdn.com/b/ntdebugging/archive/2008/05/20/understanding-ntfs-compression.aspx). *Ntdebugging Blog*. Microsoft. Retrieved 2011-03-16.
32. "Shrinking the gap: carving NTFS-compressed files" (http://www.forensicfocus.com/index.php?name=Content&pid=179). Retrieved 2011-05-29.

33. "How NTFS Works" (https://technet.microsoft.com/en-us/library/cc781134%28WS.10%29.aspx). 2003-03-28. Retrieved 2011-10-24.
34. Masiero, Manuel (2011-12-01). "Should You Compress Data On Your SSD?" (http://www.tomshardware.com/reviews/ssd-ntfs-compression,3073-11.html). *Tom's Hardware*. Bestofmedia Group. Retrieved 2013-04-05.
35. "Disk Concepts and Troubleshooting" (https://technet.microsoft.com/en-us/library/cc977213.aspx). Microsoft. Retrieved 2012-03-26.
36. "Read-Only Filegroups and Compression" (http://msdn.microsoft.com/en-us/library/ms190257.aspx). *SQL Server 2008 Books Online*. Microsoft. November 2009. Retrieved 2010-04-20.
37. "Best Practices for NTFS Compression in Windows" (http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251186). *Microsoft Knowledge Base*. Retrieved 2005-08-18.
38. "Compact OS, single-instancing, and image optimization" (https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/compact-os). Microsoft. Retrieved 1 October 2019.
39. "Sparse Files" (http://msdn.microsoft.com/en-us/library/windows/desktop/aa365564%28v=vs.85%29.aspx). *MSDN*. Microsoft. 12 October 2013. Retrieved 21 October 2013.
40. Kandoth, Suresh B. (4 March 2009). "Sparse File Errors: 1450 or 665 due to file fragmentation: Fixes and Workarounds" (http://blogs.msdn.com/b/psssql/archive/2009/03/04/sparse-file-errors-1450-or-665-due-to-file-fragmentation-fixes-and-workarounds.aspx). *CSS SQL Server Engineers*. Microsoft. Retrieved 21 October 2013.
41. "Sparse Files and Disk Quotas" (http://msdn.microsoft.com/en-us/library/aa365565.aspx). *MSDN Library*. Microsoft. 12 October 2013. Retrieved 21 October 2013.
42. "Designing a Shadow Copy Strategy" (https://technet.microsoft.com/en-us/library/cc728305.aspx). *TechNet Library*. Microsoft. 28 March 2003. Retrieved 2008-01-15.
43. cfsbloggers (July 14, 2006). "How restore points and other recovery features in Windows Vista are affected when you dual-boot with Windows XP" (http://blogs.technet.com/filecab/archive/2006/07/14/441829.aspx). *The Filing Cabinet*. Retrieved 2007-03-21.
44. "Transactional NTFS" (http://msdn2.microsoft.com/en-us/library/aa363764.aspx). *MSDN*. Microsoft. Retrieved 2007-02-02.
45. "Transactional NTFS (TxF)" (https://msdn.microsoft.com/en-us/library/windows/desktop/bb968806(v=vs.85).aspx). *Windows Dev Center (MSDN)*. Microsoft. Retrieved 24 May 2015.
46. "How Security Descriptors and Access Control Lists Work" (https://technet.microsoft.com/en-us/library/cc781716%28v=ws.10%29.aspx). *TechNet*. Microsoft. Retrieved 4 September 2015.
47. Morello, John (February 2007). "Security Watch Deploying EFS: Part 1" (https://technet.microsoft.com/en-us/magazine/2007.02.securitywatch.aspx). *Technet Magazine*. Microsoft. Retrieved 2009-01-25.
48. "How EFS Works" (https://technet.microsoft.com/library/Cc962103). *Windows 2000 Resource Kit*. Microsoft. Retrieved 25 February 2014.
49. "Chapter 18 – Choosing a File System" (https://technet.microsoft.com/library/cc750355.aspx). *MS Windows NT Workstation 4.0 Resource Guide*. Microsoft. Retrieved 25 February 2014.
50. "Naming Files, Paths, and Namespaces" (http://msdn.microsoft.com/en-us/library/aa365247%28VS.85%29.aspx#naming_conventions). *MSDN*. Microsoft. Naming Conventions. Retrieved 25 February 2014.
51. "NTFS. Partition Boot Sector" (http://www.ntfs.com/ntfs-partition-boot-sector.htm). *Ntfs.com*. Retrieved 22 September 2018.
52. "Boot Sector" (https://technet.microsoft.com/en-us/library/cc976796.aspx). *Technet.microsoft.com*. Retrieved 22 September 2018.
53. "Master File Table" (http://msdn.microsoft.com/en-us/library/windows/desktop/aa365230(v=vs.85).aspx). MSDN. July 2, 2012.
54. "NTFS Master File Table (MFT)" (http://ntfs.com/ntfs-mft.htm). *Ntfs.com*. Retrieved 22 September 2018.

55. Schwarz, Thomas. "COEN 252 Computer Forensics NTFS" (http://www.cse.scu.edu/~tschwarz/coen252_07Fall/Lectures/NTFS.html). Faculty of Organization and Informatics University of Zagreb. Retrieved May 30, 2019.
56. Since Windows XP, it is very difficult to view a listing of these files: they exist in the root directory's index, but the Win32 interface filters them out. In NT 4.0, the command line `dir` command would list the metafiles in the root directory if `/a` were specified. In Windows 2000, `dir /a` stopped working, but `dir /a \$MFT` worked.
57. "OEM Support Tools Phase 3 Service Release 2 Availability" (https://web.archive.org/web/20150223112102/http://support.microsoft.com/kb/253066/en-us). Microsoft Corporation. 2007-02-21. Archived from the original (http://support.microsoft.com/kb/253066/) on 2015-02-23. Retrieved 2010-06-16. "Windows NT File System (NTFS) File Sector Information Utility ... A tool used to dump information about an NTFS volume"
58. "The Four Stages of NTFS File Growth" (http://blogs.technet.com/b/askcore/archive/2009/10/16/the-four-stages-of-ntfs-file-growth.aspx). Retrieved 22 September 2018.
59. "A heavily fragmented file in an NTFS volume may not grow beyond a certain size" (https://support.microsoft.com/en-us/topic/a-heavily-fragmented-file-in-an-ntfs-volume-may-not-grow-beyond-a-certain-size-da1c0dfd-a5a1-90b4-4bf7-b65b13ef9d35). Archived (https://web.archive.org/web/20210506185845/https://support.microsoft.com/en-us/topic/a-heavily-fragmented-file-in-an-ntfs-volume-may-not-grow-beyond-a-certain-size-da1c0dfd-a5a1-90b4-4bf7-b65b13ef9d35) from the original on 2021-05-06. Retrieved 2021-05-19.
60. "How Oplocks function in the Windows Environment: Overview" (https://web.archive.org/web/20100823125612/http://msdn.microsoft.com/en-us/library/cc308442.aspx). Archived from the original (http://msdn.microsoft.com/en-us/library/cc308442.aspx) on 2010-08-23. Retrieved 2018-12-19.
61. "What's New in NTFS" (https://technet.microsoft.com/en-us/library/ff383236(WS.10).aspx). *Technet.microsoft.com*. Retrieved 22 September 2018.
62. "Beating the Daylight Saving Time bug and getting correct file modification times (http://www.codeproject.com/datetime/dstbugs.asp) Archived (https://web.archive.org/web/20041114032240/http://www.codeproject.com/datetime/dstbugs.asp) 2004-11-14 at the Wayback Machine" *The Code Project*
63. cfsbloggers (July 14, 2006). "How restore points and other recovery features in Windows Vista are affected when dual-booting with Windows XP" (http://blogs.technet.com/filecab/archive/2006/07/14/441829.aspx). *The Filing Cabinet*. Retrieved 2007-03-21.
64. "How to Convert FAT Disks to NTFS" (https://www.karlstechnology.com/blog/convert-fat-disks-to-ntfs/). Microsoft. Retrieved May 30, 2019.
65. "How to use Convert.exe to convert a partition to the NTFS file system" (https://www.eduhk.hk/ocio/content/faq-how-convert-fat32-file-system-ntfs-windows-xp). Microsoft Corporation. 2007-02-12. Retrieved 2010-12-26.
66. "Unable to open content synced in a OneDrive folder on an external drive" (https://support.microsoft.com/en-us/office/unable-to-open-content-synced-in-a-onedrive-folder-on-an-external-drive-d38a7d5e-c099-4b72-8d84-3f62f2efc929). *Microsoft Support*. Retrieved 2021-04-03.
67. André, Jean-Pierre (March 1, 2019). "NTFS-3G: Junction Points, Symbolic Links and Reparse Points" (https://jp-andre.pagesperso-orange.fr/junctions.html). *jp-andre.pagesperso-orange.fr*.
68. "FreeBSD 3.2 Release Notes" (https://www.freebsd.org/releases/3.2R/notes.html). 17 May 1999. Retrieved 2020-06-15.
69. "mount_ntfs - OpenBSD manual pages" (https://man.openbsd.org/mount_ntfs.8). Retrieved 2020-06-15.
70. "Announcing NetBSD 1.5" (https://www.netbsd.org/releases/formal-1.5/NetBSD-1.5.html). 6 December 2000. Retrieved 2020-06-15.
71. "OpenBSD 4.9" (http://openbsd.com/49.html). *Openbsd.com*. Retrieved 22 September 2018.
72. "About Tuxera" (https://www.tuxera.com/about-us/). Retrieved 2020-06-15.

73. Alvares, Milind (2 October 2009). "Snow Leopard's hidden NTFS read/write support" (https://web.archive.org/web/20100810084414/http://smokingapples.com/software/tutorials/snow-leopards-hidden-ntfs-readwrite-support/). Archived from the original (http://smokingapples.com/software/tutorials/snow-leopards-hidden-ntfs-readwrite-support/) on 10 August 2010. Retrieved 18 September 2010.
74. "Jan Kratochvil: Captive: The first free NTFS read/write filesystem for GNU/Linux" (http://www.jankratochvil.net/project/captive/). Retrieved 2020-06-15.
75. "OpenBSD adds fuse(4) support for adding file systems in userland" (http://undeadly.org/cgi?action=article&sid=20131108082749&mode=expanded&count=2). OpenBSD Journal. 2013-11-08. Retrieved 2013-11-08.
76. "NTFS-3G Stable Read/Write Driver" (http://www.ntfs-3g.org/). 2009-07-25.
77. "Tuxera NTFS for Mac" (http://www.tuxera.com/products/tuxera-ntfs-for-mac/). Tuxera. August 30, 2011. Retrieved September 20, 2011.
78. "NTFS for Mac OS X, communication channel between Mac OS X and Windows" (http://www.paragon-software.com/home/ntfs-mac/release.html). Paragon Software Group. Retrieved September 20, 2011.
79. Seagate Read/Write NTFS driver for Mac OS X (http://www.seagate.com/ww/v/index.jsp?locale=en-US&name=goflex-software&vgnextoid=11c1fab114b48210VgnVCM1000001a48090aRCRD) Archived (https://web.archive.org/web/20110210152004/http://www.seagate.com/ww/v/index.jsp?locale=en-US&name=goflex-software&vgnextoid=11c1fab114b48210VgnVCM1000001a48090aRCRD) 2011-02-10 at the Wayback Machine
80. "NTFS plugin for NetDrive" (https://ecsoft2.org/ntfs-plugin-netdrive). *ecsoft2.org*. Retrieved 2020-09-09.
81. "NetDrive for OS/2" (https://www.arcanoae.com/shop/netdrive-for-os2/). *arcanoae.com*. Retrieved 2020-09-09.
82. "Avira NTFS4DOS Personal" (https://web.archive.org/web/20100619161828/http://www.free-av.com/en/tools/11/avira_ntfs4dos_personal.html). Archived from the original (http://www.free-av.com/en/tools/11/avira_ntfs4dos_personal.html) on June 19, 2010. Retrieved 2009-07-25.
83. "Download Avira NTFS4DOS Personal 1.9" (http://www.softpedia.com/progDownload/Avira-NTFS4DOS-Personal-Download-104314.html). Retrieved 22 September 2018.

Further reading

- Bolosky, William J.; Corbin, Scott; Goebel, David; Douceur, John R. (January 2000). "Single Instance Storage in Windows 2000" (https://www.microsoft.com/en-us/research/publication/single-instance-storage-in-windows-2000/). *Proceedings of 4th USENIX Windows Systems Symposium*.
- Custer, Helen (1994). *Inside the Windows NT File System* (https://archive.org/details/insidewindowsntf00cust). Microsoft Press. ISBN 978-1-55615-660-1.
- Nagar, Rajeev (1997). *Windows NT File System Internals: A Developer's Guide* (https://archive.org/details/windowsntfilesys00naga). O'Reilly. ISBN 978-1-56592-249-5.
- "NTFS Technical Reference" (https://technet.microsoft.com/en-us/library/cc758691(WS.10).aspx). *Microsoft TechNet*. Microsoft. 28 March 2003.