NTFS

From Wikipedia, the free encyclopedia

NTFS (**New Technology File System**)[1] is a proprietary file system developed by Microsoft Corporation for its Windows NT line of operating systems, beginning with Windows NT 3.1 and Windows 2000, including Windows XP, Windows Server 2003, and all their successors to date.[7]

NTFS supersedes the FAT file system as the preferred file system for Microsoft Windows operating systems. NTFS has several technical improvements over FAT and HPFS (High Performance File System), such as improved support for metadata, and the use of advanced data structures to improve performance, reliability, and disk space utilization, plus additional extensions, such as security access control lists (ACL) and file system journaling.

Contents

NTFS

Developer	Microsoft
Full name	New Technology File System[1]
Introduced	July 1993 (Windows NT 3.1)
Partition identifier	`0x07` (MBR) `EBD0A0A2-B9E5-4433-87C0-68B6B72699C7` (GPT)
Structures	
Directory contents	B+ tree[2]
File allocation	Bitmap
Bad blocks	$badclus (MFT Record)
Limits	
Max. file size	16 EiB – 1 KiB (format); 16 TiB – 64 KiB (Windows 7, Windows Server 2008 R2 or earlier implementation)[3] 256 TiB – 64 KiB (Windows 8, Windows Server 2012 implementation)[4]
Max. number of files	4,294,967,295 (2^{32}-1)[3]
Max. filename length	255 UTF-16 code units[5]
Max. volume size	2^{64} clusters – 1 cluster (format); 256 TB (256 × 1024^4 bytes) – 64 KB (64 × 1024 bytes) (implementation)[3]
Allowed characters in filenames	In Posix namespace, any UTF-16 code unit (case sensitive) except U+0000 (NUL) and / (slash). In Win32 namespace, any UTF-16 code unit (case insensitive) except U+0000 (NUL) / (slash) \ (backslash) : (colon) * (asterisk) ? (Question mark) " (quote) < (less than) > (greater than) and

	| (pipe) [5]
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 streams* 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)[6]
Supported operating systems	Windows NT family (Windows NT 3.1 to Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012), Mac OS X, GNU/Linux

History

In the mid-1980s, Microsoft and IBM formed a joint project to create the next generation of graphical operating system. The result of the project was OS/2, but Microsoft and IBM disagreed on many important issues and eventually separated. OS/2 remained an IBM project. Microsoft started[*citation needed*] to work on Windows NT. The OS/2 file system HPFS contained several important new features. When Microsoft created their new operating system, they borrowed many of these concepts for NTFS.[8] Probably as a result of this common ancestry, HPFS and NTFS share the same disk partition identification type code (07). Sharing an ID is unusual, since there were dozens of available codes, and other major file systems have their own code. 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. It is also clear that NTFS owes some of its architectural design to Files-11 used by VMS. Dave Cutler was the main lead for both VMS and Windows NT.

Versions

The NTFS on-disk format has five released versions:

- v1.0 with NT 3.1,[*citation needed*] released mid-1993
- v1.1 with NT 3.5,[*citation needed*] released fall 1994
- v1.2 with NT 3.51 (mid-1995) and NT 4 (mid-1996) (occasionally referred to as "NTFS 4.0", because file system driver version is 4.0)
- v3.0 from Windows 2000 ("NTFS V5.0" or "NTFS5")[9]
- v3.1 from Windows XP (autumn 2001; "NTFS V5.1")[*citation needed*]

V1.0 and V1.1 (and newer) are incompatible: that is, volumes written by NT 3.5x cannot be read by NT 3.1 until an update on the NT 3.5x CD is applied to NT 3.1, which also adds FAT long file name support.[10] Below are descriptions of some of the versions:

- V1.2 supports compressed files, named streams, ACL-based security, etc.[2]

- V3.0 added disk quotas, encryption, sparse files, reparse points, update sequence number (USN) journaling, the $Extend folder and its files, and reorganized security descriptors so that multiple files using the same security setting can share the same descriptor.[2]
- V3.1 expanded the Master File Table (MFT) entries with redundant MFT record number (useful for recovering damaged MFT files).

Windows Vista introduced Transactional NTFS, NTFS symbolic links, partition shrinking and self-healing functionality,[11] though these features owe more to additional functionality of the operating system than the file system itself.

The NTFS.sys version (i.e. NTFS v5.0 introduced with Windows 2000) should not be confused with the on-disk NTFS format version (v3.1 since Windows XP).[12] The NTFS v3.1 on-disk format is unchanged from the introduction of Windows XP and is used in Windows Server 2003, Windows Server 2008, Windows Vista, and Windows 7. The confusion arises when no differentiation is made when features are implemented into the NTFS.sys driver within the Windows OS rather than in the NTFS on-disk format. An incident of this was when Microsoft detailed new features within NTFS in Windows 2000 and they called it NTFS v5.0, yet it is the NTFS.sys driver that is at that version and the on-disk format is only at v3.0.[9]

Features

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

NTFS Log

NTFS is a journaling file system and uses the NTFS Log ($LogFile) to record metadata changes to the volume.

It is a critical functionality of NTFS (a feature that FAT/FAT32 does not provide) for ensuring that its internal complex data structures (notably the volume allocation bitmap), or data moves performed by the defragmentation API, the 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) will remain consistent in case of system crashes, and allow easy rollback of uncommitted changes to these critical data structures when the volume is remounted.

USN Journal

The USN Journal (Update Sequence Number Journal) is a system management feature that records changes to all 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.[13] This journal can be enabled or disabled on non-system volumes[14] and is not enabled by default for a newly added drive.

Hard links and short filenames

Originally included to support the POSIX subsystem in Windows NT,[15] hard links are similar to directory junctions, but used for files instead of directories. Hard links can only be applied to files on the same volume since an additional filename record is added to the file's MFT record. Short (8.3) filenames are also implemented as additional filename records and directory entries that are linked and updated together. Hard links also have the behavior that changing the size or attributes of a file may not update the directory entries of other links until they are opened.[16]

Alternate data streams (ADS)

Alternate data streams allow more than one data stream to be associated with a filename, using the filename format "filename:streamname" (e.g., "text.txt:extrastream"). Alternate streams are not listed in Windows Explorer, and their size is not included in the file's size. Only the main stream of a file is preserved when it is copied to a FAT-formatted USB drive, attached to an e-mail, or uploaded to a website. As a result, using alternate streams for critical data may cause problems. NTFS Streams were introduced in Windows NT 3.1, to enable Services for Macintosh (SFM) to store Macintosh 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.

Malware has used alternate data streams to hide its code;[17] some malware scanners and other special tools now check for data in alternate streams. Microsoft provides a tool called Streams[18] to allow users to view streams on a selected volume.

Very small ADS (called Zone.Identifier) are also added within Internet Explorer (and now also other browsers) to mark files that have been downloaded from external sites: they may be unsafe to run locally and the local shell will require confirmation from the user before opening them.[19] When the user indicates that they no longer want this confirmation dialog, this ADS is simply dropped from the MFT entry for downloaded files.

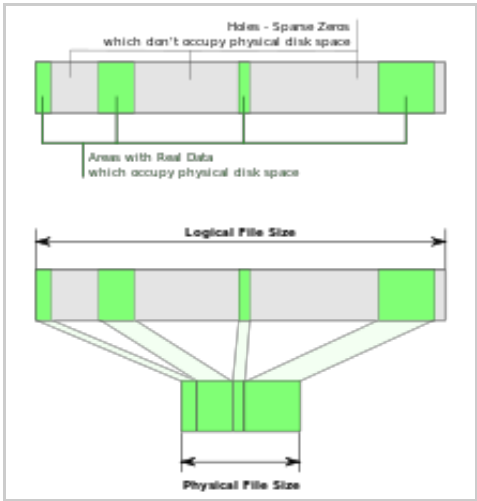
Some media players have also tried to use ADS to store custom metadata to media files, in order to organize the collections, without modifying the effective data content of the media files themselves (using embedded tags when they are supported by the media file formats such as MPEG and Ogg containers); these metadata may be displayed in the Windows Explorer as extra information columns, with the help of a registered Windows Shell extension that can parse them, but most media players prefer to use their own separate database instead of ADS for storing this information (notably because ADS are visible to all users of these files, instead of being managed with distinct per-user security settings and having their values defined according to user preferences).

Starting from PowerShell 3.0 it is possible to manage Alternate Data Streams natively with 7 cmdlets: Add-Content, Clear-Content, Get-Content, Get-Item, Out-String, Remove-Item, Set-Content.[*citation needed*]

Sparse files

Sparse files are files with segments stored at different file offsets with no actual storage space used for the space between segments. When a file is read back, the file system driver returns zeros for any data that does not actually exist, so the file may appear to be mostly filled with zeros.

Database applications, for instance, sometimes use sparse files.[20] Because of this, Microsoft has implemented support for efficient storage of sparse files by allowing an application to specify regions of empty (zero) data. An application that reads a sparse file reads it in the normal manner with the file system calculating what data should be returned based upon the file offset. As with compressed files, the actual sizes of sparse files are not taken into account when determining quota limits.[21][22]



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

File compression

NTFS can compress files using LZNT1 algorithm (a variant of the LZ77[23]). Files are compressed in 16-cluster chunks. With 4 kB clusters, files are compressed in 64 kB chunks. 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 not unreasonable random-access times - the OS just has to follow the chain of fragments. However, large compressible files become highly fragmented since every chunk <

64KB becomes a fragment.[24][25] 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 less than 900 bytes or so are stored with the directory entry in the MFT.[26]

Flash memory such as SSD drives do not have the head movement delays of hard disk drives, so fragmentation has only small effects. Users with a fast multi-core processor will find improvements in application speed by compressing their applications and data as well as a reduction in space used.[27] Note that SSDs with Sandforce controllers compress data already. However, since less data is transferred, there is a reduction in I/Os.

The best use of compression is for files that are repetitive, written seldom, usually accessed sequentially, and not themselves compressed. Log files are an ideal example. Compressing system files used at bootup like drivers, NTLDR, winload.exe, or BOOTMGR may prevent the system from booting correctly.[28] However, in later editions of Windows, compression of important system files is disallowed.

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 mostly[29] transparent, Microsoft recommends avoiding compression on server systems and/or network shares holding roaming profiles because it puts a considerable load on the processor.[30] Compression is not recommended by Microsoft for files exceeding 30 MB because of the performance hit.[*citation needed*] Since many fragments are created for compressible files, defragging may take longer.

The slowest link in a computer is not the CPU but the speed of the hard drive. NTFS compression allows the limited, slow non-RAM storage to be better used, saving space and often time.[31] (This assumes that compressed file fragments are stored consecutively.)

NTFS compression can also serve as a replacement for sparse files when a program (e.g., a download manager) is not able to create files without content as sparse files.[*citation needed*]

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 (*copy-on-write*). The old file data is overlaid on the new when the user requests a revert to an earlier version. 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.[32]

Transactional NTFS

As of Windows Vista, applications can use Transactional NTFS to group changes to files together into a transaction. The transaction will guarantee that all changes happen, or none of them do, and it will guarantee that applications outside the transaction will not see the changes until they are committed.[33]

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 filesystems, including with their local live or offline caches.

Encrypting File System (EFS)

Main article: Encrypting File System

EFS provides strong[34] 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 file header. 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.[35] 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

Main article: NTFS reparse point

This feature was introduced in NTFS v3. Reparse points are used by associating a reparse tag in the user space attribute of a file or directory. When the object manager (see Windows NT line executive) 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 call and executes its special functionality. Reparse points are used to implement Volume Mount Points, Directory Junctions, Hierarchical Storage Management, Native Structured Storage, Single Instance Storage, and Symbolic Links.[*citation needed*]

Volume mount points

Volume mount points are similar to Unix mount points, where the root of another file system is attached to a directory. In NTFS, this allows additional file systems to be mounted without requiring a separate drive letter (such as `C:` or `D:`) for each.

Once a volume has been mounted on top of an existing directory of another volume, the contents previously listed in that directory become invisible and are replaced by the content of the root directory of the mounted volume. The mounted volume could still have its own drive letter assigned separately. The file system does not allow volumes to be mutually mounted on each other. Volume mount points can be made to be either persistent (remounted automatically after system reboot) or not persistent (must be manually remounted after reboot).[*citation needed*]

Mounted volumes may use other file systems than just NTFS; notably they may be remote shared directories, possibly with their own security settings and remapping of access rights according to the remote file system policy.[*citation needed*]

Directory junctions

Main article: NTFS junction point

Directory junctions are similar to volume mount points, but reference other directories in the file system instead of other volumes. For instance, the directory `C:\exampledir` with a directory junction attribute that contains a link to `D:\linkeddir` will automatically refer to the directory `D:\linkeddir` when it is accessed by a user-mode application.[2] This function is conceptually similar to symbolic links to directories in Unix, except that the target in NTFS must always be another directory (typical Unix file systems allow the target of a symbolic link to be any type of file).

Directory joins (which can be created with the command `MKLINK /J junctionName targetDirectory` and removed with `RMDIR junctionName` from a console prompt) are persistent, and resolved on the server side as they share the same security realm of the local system or domain on which the parent volume is mounted and the same security settings for its contents as the content of the target directory; however the junction itself may have distinct security settings. Unlinking a directory junction join does not delete files in the target directory.[*citation needed*]

Note that some directory junctions are installed by default on Windows Vista, for compatibility with previous versions of Windows, such as `Documents and Settings` in the root directory of the system drive, which links to the `Users` physical directory in the root directory of the same volume. However they are hidden by default, and their security settings are set up so that the Windows Explorer will refuse to open them from within the Shell or in most applications, except for the local built-in SYSTEM user or the local Administrators group (both user accounts are used by system software installers). This additional security restriction has probably been made to avoid users of finding apparent duplicate files in the joined directories and deleting them by error, because the semantics of directory junctions is not the same as hardlinks; the reference counting is not used on the target contents and not even on the referenced container itself.[*citation needed*]

Directory junctions are soft links (they will persist even if the target directory is removed), working as a limited form of symbolic links (with an additional restriction on the location of the target), but it is an optimized version allowing faster processing of the reparse point with which they are implemented, with less overhead than the newer NTFS symbolic links, and can be resolved on the server side (when they are found in remote shared directories).[*citation needed*]

Symbolic links

Main article: NTFS symbolic link

Symbolic links (or soft links) were introduced in Windows Vista.[36] Symbolic links are resolved on the client side. So when a symbolic link is shared, the target is subject to the access restrictions on the client, and not the server.[*citation needed*]

Symbolic links can be created either to files (created with `MKLINK symLink targetFilename`) or to directories (created with `MKLINK /D symLinkD targetDirectory`), but (unlike Unix symbolic links) the semantic of the link must be provided with the created link. The target however need not exist or be available when the symbolic link is created: when the symbolic link will be accessed and the target will be checked for availability, NTFS will also check if it has the correct type (file or directory); it will return a not-found error if the existing target has the wrong type.[*citation needed*]

They can also reference shared directories on remote hosts or files and subdirectories within shared directories: their target is not mounted immediately at boot, but only temporarily on demand while opening them with the OpenFile() or CreateFile() API. Their definition is persistent on the NTFS volume where they are created (all types of symbolic links can be removed as if they were files, using `DEL symLink` from a command line prompt or batch).[*citation needed*]

Distributed Link Tracking (DLT)

See also: File shortcut

Distributed link tracking allows applications to track files, shell shortcuts or OLE links even if they were renamed or moved to another volume within the same machine, domain or workgroup.[37] Tracking is implemented as a system service, which uses the object identifier (OID) index stored in a metafile.[38] When the application requests a track to a file or directory, the tracking service creates the OID entry, which points to the file, and file rename, copy or move operation to a NTFS v3 volume also copies the object ID. This allows the tracking service to eventually find the target file.

Single Instance Storage (SIS)

When there are several directories that have different but similar files, some of these files may have identical content. Single instance storage allows identical files to be merged to one file and create references to that merged file. SIS consists of a file system filter that manages copies, modification and merges to files; and a user space service (or *groveler*) that searches for files that are identical and need merging. SIS was mainly designed for remote installation servers as these may have multiple installation images that contain many identical files; SIS allows these to be consolidated but, unlike for example hard links, each file remains distinct; changes to one copy of a file will leave others unaltered. This is similar to copy-on-write, which is a technique by which memory copying is not really done until one copy is modified.[39]

Hierarchical Storage Management (HSM)

Hierarchical Storage Management is a means of transferring files that are not used for some period of time to less expensive storage media. When the file is next accessed, the reparse point on that file determines that it is needed and retrieves it from storage.[*citation needed*]

Native Structured Storage (NSS)

NSS was an ActiveX document storage technology that has since been discontinued by Microsoft.[*citation needed*] It allowed ActiveX Documents to be stored in the same multi-stream format that ActiveX uses internally. An NSS file system filter was loaded and used to process the multiple streams

transparently to the application, and when the file was transferred to a non-NTFS formatted disk volume it would also transfer the multiple streams into a single stream.[40]

Interoperability

Details on the implementation's internals are not released, which makes it difficult for third-party vendors to provide tools to handle NTFS.

Microsoft Windows

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.[41]

Mac OS X

Mac OS X 10.3 and later include read-only support for NTFS-formatted partitions. The GPL-licensed NTFS-3G also works on Mac OS X through FUSE and allows reading and writing to NTFS partitions. A performance enhanced commercial version, called *Tuxera NTFS for Mac*,[42] is also available from the NTFS-3G developers. Paragon Software Group sells a read-write driver named *NTFS for Mac OS X*,[43] which is also included on some models of Seagate hard drives.[44] Native NTFS write support has been discovered in Mac OS X 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.[45]

Linux

The ability to read and write to NTFS is provided by the NTFS-3G driver. It is included in most Linux distributions. Other solutions exist as well:

- Linux kernel 2.2: Kernel versions 2.2.0 and later include the ability to read NTFS partitions
- Linux kernel 2.6: Kernel versions 2.6.0 and later contain a driver written by Anton Altaparmakov (University of Cambridge) and Richard Russon. It supports file read, overwrite and resize.
- NTFSMount: A read/write userspace NTFS driver. It provides read-write access to NTFS, excluding writing compressed and encrypted files, changing file ownership, and access rights.[46]
- Tuxera NTFS: High-performance read/write commercial kernel driver, mainly targeted for embedded devices from Tuxera, which also develops the open source NTFS-3G driver.
- NTFS for Linux: A commercial driver with full read/write support is available as free and non-free download(s) for Desktop and Embedded Linux systems from Paragon Software Group.
- Captive NTFS (discontinued):[47] A 'wrapping' driver that uses Windows' own driver, `ntfs.sys`.

Note that all three userspace drivers, namely NTFSMount, NTFS-3G and Captive NTFS, are built on the Filesystem in Userspace (FUSE), a Linux kernel module tasked with bridging userspace and kernel code to save and retrieve data. All drivers listed above (except Tuxera NTFS and Paragon NTFS for Linux) are open source (GPL). Due to the complexity of internal NTFS structures, both the built-in 2.6.14 kernel driver and the FUSE drivers disallow changes to the volume that are considered unsafe, to avoid corruption.

Others

eComStation, and FreeBSD offer read-only NTFS support (there is a beta NTFS driver that allows write/delete for eComStation, but is generally considered unsafe). A free third-party tool for BeOS, which was based on NTFS-3G, allows full NTFS read and write. NTFS-3G also works on Mac OS X, FreeBSD, NetBSD, Solaris, QNX and Haiku, in addition to Linux, through FUSE.[48] A free for personal use read/write driver for MS-DOS called "NTFS4DOS" also exists.[49][50] 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. OpenBSD offer read-only NTFS support by default on i386 and amd64 platforms as of version 4.9 released 1. May 2011.[51]

Conversion from other file systems

Microsoft provides a tool (convert.exe) to convert to NTFS from other file systems. Supported systems include HPFS (only on Windows NT 3), FAT16 and, on Windows 2000 and later, FAT32.[52][53]

Resizing

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

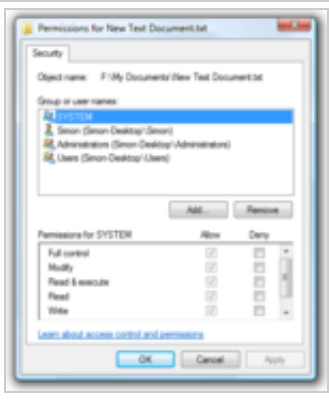
Universal time

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. However, Windows NT and its descendants keep internal timestamps as UTC and make the appropriate conversions for display purposes. Therefore, NTFS timestamps are in UTC. 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.[54]

Internals

Internally, NTFS uses B+ trees to index file system data. Although complex to implement, this allows faster file look up times in most cases. 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.[55]

NTFS allows any sequence of 16-bit values for name encoding (file names, stream names, index names, etc.) except 0x0000. This means UTF-16 codepoints 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).



NTFS filesystem permissions on a Windows Vista system.

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. This abstract approach allowed easy addition of file system features during Windows NT's development—an interesting example is the addition of fields for indexing used by the Active Directory software. This also enables software like Everything or Ultrasearch[56] to perform instantaneous real-time searches for file and folder names, without relying on an indexing service.

The MFT structure supports algorithms which minimize disk fragmentation.[57] A directory entry consists of a filename and a "file ID", 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.

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. 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.

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. 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. $Boot is usually 8192 bytes long.[*citation needed*]
8	`$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 $Bad and contains all clusters on the volume not in the first stream.[58]
9	`$Secure`	Access control list database that reduces overhead having many identical ACLs stored with each file, by uniquely storing these ACLs in this database only (contains two indices: $SII *(Standard_Information ID)* and $SDH *(Security Descriptor Hash)*, which index the stream named $SDS containing actual ACL table).[2]
10	`$UpCase`	A table of unicode uppercase characters for ensuring case insensitivity in Win32 and DOS namespaces.
11	`$Extend`	A filesystem directory containing various optional extensions, such as $Quota, $ObjId, $Reparse or $UsnJrnl.
12 ... 23	Reserved for $MFT extension entries.[59]	
usually 24	`$Extend\$Quota`	Holds disk quota information. Contains two index roots, named $O and $Q.
usually 25	`$Extend\$ObjId`	Holds link tracking information. Contains an index root and allocation named $O.
usually 26	`$Extend\$Reparse`	Holds reparse point data (such as symbolic links). Contains an index root and allocation named $R.
27 ...	*`file.ext`*	Beginning of regular file entries.

These metafiles are treated specially by Windows and are difficult to directly view: special purpose-built tools are needed.[60] One such tool is the nfi.exe ("NTFS File Sector Information Utility") that is freely distributed as part of the Microsoft "OEM Support Tools".[61]

From MFT records to attribute lists, attributes, and streams

For each file (or directory) described in the MFT record, there's a linear repository of stream descriptors (also named *attributes*), packed together in a variable-length record (also named an *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 stream (or *attribute*) itself has a single type (internally just a fixed-size integer in the stored descriptor, but most often handled in applications using an equivalent symbolic name in the `FileOpen()` or `FileCreate()` API call), a single optional stream name (completely unrelated to the effective filenames), plus optional associated data for that stream. For NTFS, the standard data of files, or the index data for directories are handled the same way as other data for alternate data streams, or for standard attributes. They are just one of the attributes stored in one or several attribute lists.

- For each file described in the MFT record (or in the non-resident repository of stream descriptors, see below), the stream descriptors identified by their (stream type value, stream name) must be unique. Additionally, NTFS has some ordering constraints for these descriptors.
- There's a predefined null stream type, used to indicate the end of the list of stream descriptors in the streams repository for that file. It must be present as the last stream descriptor in each stream repository (all other storage space available after it will be ignored and just consists in padding bytes to match the record size in the MFT or a cluster size in a non-resident streams repository).
- Some stream types are required and must be present in each MFT record, except unused records that are just indicated by a stream with null stream type.
    - This is the case for the standard attributes that are stored as a fixed-size record and containing the timestamps and other basic single-bit attributes (compatible with those managed by FAT/FAT32 in DOS or Windows 95/98 applications).
- Some stream types cannot have a name and must remain anonymous.
    - This is the case for the standard attributes, or for the preferred NTFS "filename" stream type, or the

"short filename" stream type, when it is also present (for compatibility with DOS-like applications, see below). It is also possible for a file to only contain a short filename, in which case it will be the preferred one, as listed in the Windows Explorer.
    - The filename streams stored in the streams repository do not make the file immediately accessible through the hierarchical filesystem. In fact, all the filenames must be indexed separately in at least one separate directory on the same volume, with its own MFT entry and its own security descriptors and attributes, that will reference the MFT entry number for that 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 $DATA but with an anonymous name, and the ADS's are similar but must be named.
- On the opposite, the default data stream of directories has a distinct type, but are not anonymous: they have a stream name ("$I30" in NTFS 3+) that reflects its indexing format.

All streams 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".[62]

Resident vs. non-resident data streams

To optimize the storage and reduce the I/O overhead for the very common case of streams with very small associated data, NTFS prefers to place this data within the stream descriptor (if the size of the stream descriptor does not then exceed the maximum size of the MFT record or the maximum size of a single entry within an non-resident stream repository, see below), instead of using the MFT entry space to list clusters containing the data; in that case, the stream descriptor will not store the data directly but will just store an allocation map pointing to the actual data stored elsewhere on the volume.[63] When the stream data can be accessed directly from within the stream descriptor, 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 stream descriptors (such as the preferred filename, the basic file attributes, or the main allocation map for each non-resident stream) cannot be made non-resident.
- Encrypted-by-NTFS, sparse data streams, or compressed data streams cannot be made resident.
- The format of the allocation map for non-resident streams depends on its capability of supporting sparse data storage. In the current implementation of NTFS, once a non-resident stream data 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 data stream is too much fragmented, so that its effective allocation map cannot fit entirely within the MFT record, the allocation map may be also stored as an non-resident stream, with just a small resident stream containing the indirect allocation map to the effective non-resident allocation map of the non-resident data stream.
- When there are too many streams for a file (including ADS's, extended attributes, or security descriptors), so that their descriptors cannot fit all within the MFT record, a non-resident stream may also be used to store an additional repository for the other stream descriptors (except those few small streams that cannot be non-resident), using the same format as the one used in the MFT record, but without the space constraints of the MFT record.

The NTFS filesystem driver will sometimes attempt to relocate the data of some of these non-resident streams into the streams repository, and will also attempt to relocate the stream descriptors stored in a non-resident repository back to the stream repository of 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 × 19,526,158 = 78,104,632 resident files.

Opportunistic locks

Opportunistic locks (oplocks) allow clients to alter their buffering strategy for a given file or stream in order to increase performance and reduce network use.[64] 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.[65]

Limitations

The following are a few limitations of NTFS:

Compression
: 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.[66]

Maximum cluster size
: The maximum cluster size is 64 kB.[67]

File names
: File names are limited to 255 UTF-16 code points. 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`.[3] (dot) and $Extend are both directories; the others are files. The NT kernel limits full paths to 32,767 UTF-16 code points.
Some other file-naming limitations and recommendations can be found on http://msdn.microsoft.com/en-us/library/aa365247%28VS.85%29.aspx#naming_conventions .

Maximum volume size
: In theory, the maximum NTFS volume size is $2^{64}-1$ clusters. However, the maximum NTFS volume size as implemented in Windows XP Professional is $2^{32}-1$ clusters partly due to partition table limitations. For example, using 64 kB clusters, the maximum Windows XP NTFS volume size is 256 TBs minus 64 KBs. 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 lifted in Windows XP SP1.) Because partition tables on master boot record (MBR) disks only support partition sizes up to 2 TB, dynamic or GPT volumes must be used to create NTFS volumes over 2 TB. Booting from a GPT volume to a Windows environment requires a system with UEFI and 64-bit support.[68]

Maximum file size

As designed, the maximum NTFS file size is 16 EB (16×1024^6 or 2^{64} bytes) minus 1 kB or 18,446,744,073,709,550,592 bytes. As implemented, the maximum NTFS file size is 16 TB minus 64 kB or 17,592,185,978,880 bytes. With Windows 8 and Windows Server 2012, the maximum NTFS file size is 256 TB minus 64 KB or 281,474,976,645,120 bytes.[4]

Alternate data streams

Windows system calls may handle alternate data streams.[3] Depending on the operating system, utility and remote file system, a file transfer might silently strip data streams.[3] 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.[3]

Developers

NTFS developers include:[69]

- Tom Miller
- Gary Kimura
- Brian Andrew
- David Goebel

See also

- Comparison of file systems
- Files-11: ODS-2 has similarities to NTFS (compare `INDEXF.SYS` and `$Mft`, and `BITMAP.SYS` and `$Bitmap`, for examples)
- HPFS, file system created for the OS/2 operating system
- NTFSDOS (Winternals' implementations and tools for DOS)
- ntfsresize
- Samba (software)
- ReFS

References

1. ^ *a b* "Windows XP: Format backup drives using NTFS" (http://www.microsoft.com/windowsxp/using/setup/tips/advanced/ntfs.mspx). Microsoft. September 7, 2006.
2. ^ *a b c d e* Mark Russinovich. "Inside Win2K NTFS, Part 1" (http://msdn2.microsoft.com/en-us/library/ms995846.aspx). Microsoft Developer Network. Retrieved 2008-04-18.
3. ^ *a b c d e f g* Microsoft TechNet (2003-03-28). "How NTFS Works" (http://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx). *Windows Server 2003 Technical Reference*. Retrieved 2011-09-12.
4. ^ *a b* "6 Appendix A: Product Behavior" (http://msdn.microsoft.com/en-us/library/ff469400%28v=prot.20%29.aspx#id116). Retrieved 2012-09-21.
5. ^ *a b* Richard Russon and Yuval Fledel. "NTFS Documentation" (http://dubeyko.com/development/FileSystems/NTFS/ntfsdoc.pdf). Retrieved 2011-06-26.
6. **^** Rick Vanover. "Windows Server 8 data deduplication" (http://www.techrepublic.com/blog/datacenter/windows-server-8-data-deduplication-what-you-need-to-know/4887). Retrieved 2011-12-02.
7. **^** Custer, Helen (1994). *Inside the Windows NT File System*. Microsoft Press. ISBN 978-1-55615-660-1.
8. **^** Kozierok, Charles M. (April 17, 2001). "Overview and History of NTFS" (http://www.pcguide.com/ref/hdd/file/ntfs/over-c.html). PCGuide.
9. ^ *a b* "Inside Win2K NTFS, Part 1" (http://msdn.microsoft.com/en-us/library/ms995846.aspx). Microsoft. January 26, 2011.
10. **^** "Recovering Windows NT After a Boot Failure on an NTFS Drive" (http://support.microsoft.com/kb/q129102/).

Microsoft. November 1, 2006.
11. ^ 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 Corporation. pp. 14–20. Retrieved 2007-09-04.
12. ^ "New Capabilities and Features of the NTFS 3.1 File System" (http://support.microsoft.com/kb/310749). Microsoft. December 1, 2007.
13. ^ "Change Journals (Windows)" (http://msdn.microsoft.com/en-us/library/aa363798.aspx). MSDN. Retrieved 2010-04-16.
14. ^ "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.
15. ^ MS Windows NT Workstation 4.0 Resource Guide, "POSIX Compatibility (http://www.microsoft.com/technet/archive/ntwrkstn/reskit/poscomp.mspx?mfr=true)"
16. ^ MSDN (http://msdn.microsoft.com/en-us/library/aa365006%28VS.85%29.aspx)
17. ^ Malware utilising Alternate Data Streams? (https://www.auscert.org.au/render.html?it=7967), AusCERT Web Log, 21 August 2007
18. ^ Sysinternals Streams v1.56 (http://technet.microsoft.com/en-us/sysinternals/bb897440.aspx)
19. ^ 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"
20. ^ Sparse File Errors: 1450 or 665 due to file fragmentation: Fixes and Workarounds (http://blogs.msdn.com/psssql/archive/2009/03/04/sparse-file-errors-1450-or-665-due-to-file-fragmentation-fixes-and-workarounds.aspx), Microsoft Customer Service and Support (CSS) SQL Server Engineers Blog, March 4, 2009
21. ^ "Sparse Files" (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/sparse_files.asp). MSDN Platform SDK: File Systems. Retrieved 2005-05-22.
22. ^ "Sparse FIles and Disk Quotas" (http://msdn2.microsoft.com/en-us/library/aa365565.aspx). Win32 and COM Development: File Systems. Retrieved 2007-12-05.
23. ^ "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.
24. ^ "Understanding NTFS Compression" (http://blogs.msdn.com/b/ntdebugging/archive/2008/05/20/understanding-ntfs-compression.aspx). Retrieved 2011-03-16.
25. ^ "Shrinking the gap: carving NTFS-compressed files" (http://www.forensicfocus.com/index.php?name=Content&pid=179). Retrieved 2011-05-29.
26. ^ "How NTFS Works" (http://technet.microsoft.com/en-us/library/cc781134%28WS.10%29.aspx). 2003-03-28. Retrieved 2011-10-24.
27. ^ "Should You Compress Data On Your SSD?" (http://www.tomshardware.com/reviews/ssd-ntfs-compression,3073-11.html). 2011-12-01. Retrieved 2013-04-05.
28. ^ "Disk Concepts and Troubleshooting" (http://technet.microsoft.com/en-us/library/cc977213.aspx). Microsoft. Retrieved 2012-03-26.
29. ^ "Read-Only Filegroups and Compression" (http://msdn.microsoft.com/en-us/library/ms190257.aspx). SQL Server 2008 Books Online (November 2009). Retrieved 2010-04-20.
30. ^ "Best practices for NTFS compression in Windows (http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251186)." *Microsoft Knowledge Base*. Retrieved on 2005-08-18.
31. ^ Daily, Sean (January 1998). "Optimizing Disks" (http://www.windowsitlibrary.com/Content/435/07/8.html). IDG books. Retrieved 2007-12-17.
32. ^ "Designing a Shadow Copy Strategy" (http://technet.microsoft.com/en-us/library/cc728305.aspx). Microsoft TechNet. Retrieved 2008-01-15.
33. ^ "Transactional NTFS" (http://msdn2.microsoft.com/en-us/library/aa363764.aspx). MSDN. Retrieved 2007-02-02.
34. ^ Morello, John (February 2007). "Security Watch Deploying EFS: Part 1" (http://technet.microsoft.com/en-us/magazine/2007.02.securitywatch.aspx). Technet Magazine. Retrieved 2009-01-25.
35. ^ How EFS Works (http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distrib/dsck_efs_duwf.mspx), *Microsoft Windows 2000 Resource Kit*
36. ^ "Symbolic Links (Windows)" (http://msdn.microsoft.com/en-us/library/aa365680(VS.85).aspx). MSDN.
37. ^ http://msdn.microsoft.com/en-us/library/windows/desktop/aa363997.aspx
38. ^ http://technet.microsoft.com/en-us/library/cc736811(WS.10).aspx
39. ^ "Single Instance Storage in Windows 2000" (http://research.microsoft.com/apps/pubs/default.aspx?id=74261) (PDF). Microsoft Research and Balder Technology Group.
40. ^ Saville, John (date unknown). *What is Native Structured Storage?* Windows IT Pro. Retrieved from http://www.windowsitpro.com/Article/ArticleID/13785/13785.html.

41. ^ 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.
42. ^ "Tuxera NTFS for Mac" (http://www.tuxera.com/products/tuxera-ntfs-for-mac/). Tuxera. August 30, 2011. Retrieved September 20, 2011.
43. ^ "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.
44. ^ 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)
45. ^ Alvares, Milind (Friday, October 2, 2009). "Snow Leopard's hidden NTFS read/write support" (http://smokingapples.com/software/tutorials/snow-leopards-hidden-ntfs-readwrite-support/). Retrieved 18 September 2010.
46. ^ "ntfsmount wiki page on linux-ntfs.org (http://wiki.linux-ntfs.org/doku.php?id=ntfsmount)"
47. ^ http://www.jankratochvil.net/project/captive/
48. ^ "NTFS-3G Stable Read/Write Driver" (http://www.ntfs-3g.org/). 2009-07-25.
49. ^ Avira NTFS4DOS Personal (http://web.archive.org/web/20100619161828/http://www.free-av.com/en/tools/11/avira_ntfs4dos_personal.html) at the Wayback Machine (archived June 19, 2010)
50. ^ Download of NTFS4DOS.EXE NTFS driver for DOS at Softpedia.com (http://www.softpedia.com/progDownload/Avira-NTFS4DOS-Personal-Download-104314.html)
51. ^ http://openbsd.com/49.html
52. ^ "How to Convert FAT Disks to NTFS" (http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/convertfat.mspx). Microsoft Corporation. 2001-10-25. Retrieved 2007-08-27.
53. ^ "How to use Convert.exe to convert a partition to the NTFS file system" (http://support.microsoft.com/kb/214579). Microsoft Corporation. 2007-02-12. Retrieved 2010-12-26.
54. ^ "Beating the Daylight Savings Time bug and getting correct file modification times (http://www.codeproject.com/datetime/dstbugs.asp)" *The Code Project*
55. ^ "Microsoft TechNet Resource Kit (http://www.microsoft.com/technet/archive/ntwrkstn/reskit/file_sys.mspx?mfr=true)"
56. ^ "UltraSearch v1.7 – Freeware for Ultra-Fast File Search" (http://www.jam-software.com/ultrasearch/).
57. ^ "Master File Table" (http://msdn.microsoft.com/en-us/library/windows/desktop/aa365230(v=vs.85).aspx). MSDN. July 2, 2012.
58. ^ $BadClust:$Bad appears to be treated specially in that its allocation of clusters doesn't cause a "multiple cluster reference" error from chkdsk
59. ^ 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.
60. ^ 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.
61. ^ For example to obtain information on the "$MFT"-Master File Table Segment the following command is used: `nfi.exe c:\$MFT` "OEM Support Tools Phase 3 Service Release 2 Availability" (http://support.microsoft.com/kb/253066/). Microsoft Corporation. 2007-02-21. Retrieved 2010-06-16. "Windows NT File System (NTFS) File Sector Information Utility [...] A tool used to dump information about an NTFS volume"
62. ^ "OEM Support Tools Phase 3 Service Release 2 Availability" (http://support.microsoft.com/kb/253066/). Microsoft Corporation. 2007-02-21. Retrieved 2010-06-16. "Windows NT File System (NTFS) File Sector Information Utility [...] A tool used to dump information about an NTFS volume"
63. ^ blogs.technet.com: Jeff Hughes, The Four Stages of NTFS File Growth (2009) (http://blogs.technet.com/b/askcore/archive/2009/10/16/the-four-stages-of-ntfs-file-growth.aspx)
64. ^ http://msdn.microsoft.com/en-us/library/cc308442.aspx
65. ^ http://technet.microsoft.com/en-us/library/ff383236(WS.10).aspx
66. ^ "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.
67. ^ "[MS-FSA]: File System Algorithms. Appendix A: Product Behavior" (http://msdn.microsoft.com/en-us/library/ff469400%28v=prot.20%29.aspx#id2). Microsoft. Retrieved 2012-01-10.
68. ^ http://www.rodsbooks.com/gdisk/booting.html
69. ^ Custer, Helen (1994). *Inside the Windows NT File System*. Microsoft Press. p. vii. ISBN 978-1-55615-660-1.

Further reading

- Bolosky, William J.; Corbin, Scott; Goebel, David; & Douceur, John R. (date). *Single Instance Storage in Windows 2000* (http://research.microsoft.com/pubs/74261/WSS2000.pdf) (PDF). Microsoft Research & Balder Technology Group, Inc. Retrieved 2010-01-22.
- Custer, Helen (1994). *Inside the Windows NT File System*. Microsoft Press. ISBN 978-1-55615-660-1.
- Nagar, Rajeev (1997). *Windows NT File System Internals: A Developer's Guide*. O'Reilly. ISBN 978-1-56592-249-5.

External links

- Documentation:
  - Microsoft: NTFS Technical Reference (http://technet.microsoft.com/en-us/library/cc758691(WS.10).aspx)
  - Microsoft: Fsutil file (http://technet.microsoft.com/en-us/library/cc788058(WS.10).aspx) (documentation for line command useful for manipulating or creating files in an NTFS volume from Windows)
  - LSoft Technologies: NTFS.com (http://www.ntfs.com/) (documentation and resources)
- Implementations and tools:
  - Tuxera (http://www.tuxera.com/) (developer of implementations and tools for various platforms)
  - NTFS-3G Safe Read/Write NTFS Driver project (http://sourceforge.net/projects/ntfs-3g/) (read/write NTFS driver for Linux, Mac OS X, OpenSolaris, FreeBSD, NetBSD, QNX, Windows and Haiku)
  - Linux-Faqs.com: NTFS FAQ (Frequently Asked Questions) (http://www.linux-faqs.com/faq/misc/ntfs.php)
  - Jan Kratochvil: Captive NTFS (http://www.jankratochvil.net/project/captive/) (a shim that used the Windows NTFS driver to access NTFS file systems under Linux)
  - TZWorks LLC: Windows Journal Parser (jp) (http://www.tzworks.net/prototype_page.php?proto_id=5) (tool for parsing the Windows NTFS Change Log Journal on live systems)
  - Graphic Engine for NTFS Analysis (gena) (http://www.tzworks.net/prototype_page.php?proto_id=28) (GUI to view NTFS internals/extract data on live systems).

Categories: 1993 software | Compression file systems | Windows disk file systems | Windows components

---