NTFS

维基百科，自由的百科全书

NTFS（英语：**New Technology File System**），是Microsoft公司开发的专用文件系统，从Windows NT 3.1开始成为Windows NT家族的默认文件系统。[4]

NTFS取代FAT（文件分配表）和HPFS（高性能文件系统）并进行一系列改进成为更加完善的安全系统，例如增强对元数据的支持，使用更高级的数据结构以提升性能、可靠性和磁盘空间利用率，并附带一系列增强功能，如访问控制列表（ACL）和文件系统日志。

其他台式机和服务器操作系统也支持NTFS。Linux和BSD提供自由及开放源代码的软件NTFS-3G，可用于读写NTFS文件。Mac OS X内核能对NTFS进行有限的读操作。

目录

历史

版本

功能

- 可伸缩性
- 日志
- 硬链接
- 可选数据流（ADS）
- 文件压缩
- 稀疏文件
- 卷影复制
- 事务
- 安全
- 加密
- 限额
- 重解析点
- 卷加载点
- 目录连接
- 符号链接
- 分层存储管理（HSM）
- 本机结构存储（NSS）
- 分布链接跟踪（DLT）
- 单实例存储（SIS）

内部实现

- 分区引导扇区
- 主文件表（MFT）
- 元文件
- 从MFT到属性、属性表和流
- 常驻文件和非常驻文件
- 机会锁
- 时间

互操作性

- Microsoft Windows
- Mac OS X
- Linux
- 其他
- 调整大小

NTFS	
开发者	Microsoft
全称	New Technology File System
发布	1993年7月 (Windows NT 3.1)
分区标识	主引导记录（MBR）:0x07 全局唯一标识分区表（GPT）:EBD0A0A2-B9E5-4433-87C0-68B6B72699C7（基本数据分区（BDP））
	结构
目录内容	B+树[1]
文件分配	位图
坏块	$BadClus（MFT记录）
	限制
最大文件尺寸	理论：16EiB-1KiB[2] 实际：16TiB-64KiB(Windows 7、Windows Server 2008 R2及以前版本)，256TiB-64KiB（Windows 8、Windows Server 2012及以后版本）[2]
最大文件数量	4,294,967,295（2^{32}-1）[2]
最长文件名	255个UTF-16编码单元[3]
最大卷容量	理论：2^{64}-1簇[2] 实际：256 TiB-64 KiB[2]
文件名字符集	在POSIX名字空间中，非U+0000（NUL）和/（斜杠）的任何UTF-16码元（大小写敏感）。在Win32名字空间中，非U+0000（NUL）、/（斜杠）、\（反斜杠）、:（冒号）、*（星号）、?（问号）、"（引号）、<（小于号）、>（大于号）和|（竖线或管道符）的任何UTF-16字符（大小写不敏感）。[3]
	功能
日期记录	创建、修改、POSIX更改、访问
日期范围	1601年1月1日– 60056年

另请参阅
参考文献
外部链接

历史

20世纪80年代中期，微软（Microsoft）和IBM合作，希望创建下一代的图形操作系统。该项目的成果为OS/2，但由于微软和IBM在很多重要问题上无法达成共识，最后合作被终止，目前OS/2至今仍属于IBM，而Microsoft从此后开始研究Windows NT。

OS/2的文件系统HPFS包含许多重要功能，当Microsoft开始创建他们自己的新操作系统时，NTFS文件系统的很多功能正是从HPFS中借鉴改善的。[5]可能正是因为他们来自于同一个项目，HPFS和NTFS使用相同的磁盘分区标识代码（0x07）。这是一种非常特殊的情况，因为可用的标识码并不匮乏，其它每个文件系统具有自己的标识码，例如，FAT拥有超过九个编号（FAT12、FAT16、FAT32等等各自都拥有不同的标识码）。这种特例也导致之后用于区分文件系统的算法当遇到代码0x07时候不得不进行额外的检测。

NTFS的开发者包括：Tom Miller、Gary Kimura、Brian Andrew以及David Goebel。

版本

微软正式发布的NTFS版本有五个：

- v1.0，随Windows NT 3.1在1993年中旬发布[4]。v1.0和v1.1和之后的所有版本不兼容。也即使用NT 3.5x写入的卷无法被NT 3.1读取。该问题的一个解决方案是使用NT 3.5x光盘更新NT 3.1，并添加对FAT系统的长文件名支持。[6]
- v1.1，随Windows NT 3.5于1994年秋季发布。
- v1.2，随Windows NT 3.51于1995年中旬发布。支持压缩文件、命名流、基于ACL（访问控制列表）的安全性等功能。[1]
- v3.0，随Windows 2000发布。支持磁盘限额、加密、稀疏文件、重解析点，更新序列数（USN）日志、$Extend文件夹（及其中的文件），并改进安全描述符设计方案，允许使用同样的安全设置的多个文件共享一个安全描述符。[1]
- v3.1随Windows XP于2001年秋季发布。在MFT中提供冗余MFT记录数扩展项，可用于恢复受损的MFT文件。

不应将NTFS.sys文件版本（如Windows 2000中的 v5.0）和NTFS磁盘格式版本（如Windows XP开始的v3.1）相混淆。[7]

	5月28日（文件时间为64位数字，计时间隔为100纳秒（每秒一千万次），从1601年开始持续58000+年）
日期分辨率	100ns
岔流	是（请参见下面的可选数据流）
属性	只读、隐藏、系统、存档、内容未索引、脱机、临时
文件系统权限	访问控制列表（ACL）
透明压缩	对每个文件，LZ77（Windows NT 3.51以上）
透明加密	对每个文件 DESX（Windows 2000以上） Triple DES（Windows XP以上） 高级加密标准（AES）（Windows XP Service Pack 1、Windows Server 2003以上）
单一实例存储（SIS）	是
操作系统支持	Windows NT 3.1以上 OSX 10.3以上（只读） Linux内核版本2.6以上 Linux内核版本2.2-2.4（只读） FreeBSD NetBSD Chrome OS Solaris ReactOS（只读）

后续的Windows的版本更新增加了许多文件系统相关的功能，但并没有改变NTFS本身。例如Windows Vista增加了NTFS符号链接、事务NTFS、磁盘收缩和自我修复，但除了符号链接外其他功能其实都由操作系统实现。

功能

相对于之前的版本，NTFS v3.0包含若干新功能：磁盘使用限额、稀疏文件支持、重解析点、分布链接跟踪，以及文件级加密（即“加密文件系统（EFS）”）。

可伸缩性

理论上，NTFS卷的最大容量为2^{64}-1簇。在Windows XP专业版中，由于分区表限制，实际实现的最大容量为2^{32}-1簇。例如，在当簇的从大小为64 KiB时，Windows XP的NTFS卷的最大容量为256 TiB减去64 KiB；而当簇大小为默认的4 KiB时，卷最大容量将变为16 TiB减去4 KiB（但这都超过了Windows XP SP1对磁盘容量的128 GiB限制）。由于主引导记录（MBR）分区表最大支持单个分区容量为2 TiB，因此如果要创建超过2 TiB的NTFS卷，必须要使用动态卷或者GPT卷。注意：微软默认的引导程序必须使用UEFI和64位操作系统才能从GPT卷引导系统。

日志

NTFS是一个日志文件系统，使用NTFS日志（$Logfile）记录卷更改元数据。这是NTFS一个非常关键的功能（FAT/FAT32不提供此项功能），可确保其内部的复杂数据结构（如比较重要的如卷分配图、磁盘碎片整理API产生的数据转移操作、MFT（主文件表）记录的更改情况（包括移动MFT记录中存储的变长属性和属性表等））和索引（在目录和安全描述符中使用）即使在系统崩溃后仍然能保证一致性，而当在卷被重新加载后，可以非常容易地回滚这些关键数据的意外修改。

USN日志是一项系统管理功能，用于记录卷中所有文件、数据流、目录的内容、属性以及各项安全设置的更改情况。应用程序可以利用日志追踪卷的更改。[8]对于非系统卷，可以选择打开或关闭日志。[9]当添加一个新卷后，默认情况下日志功能处于打开状态。

硬链接

硬链接可用于将不同的文件名直接关联到同样的文件内容。

硬链接类似于目录连接，但必须引用到文件。硬链接只能连接到同一个卷内的文件，因为每个卷拥有自己的主文件表（MFT）。硬链接有自己的元数据，因此如果更改某个硬链接的文件大小或尺寸，其他硬链接在被打开前可能不会自动更新这些信息。

硬链接原本用于支持Windows NT的POSIX子系统[10]。

Windows使用硬链接实现对8.3文件名的支持。操作系统需要该功能，因为有些古老的应用程序只能使用短文件名。NTFS将会为文件和目录创建额外的NTFS记录，但他们将总是自动同步更新（常规硬链接并不会同步更新）。

NTFS文件系统限制单个文件只能关联到1024个硬链接。[11]

可选数据流（ADS）

可选数据流使单个文件可以关联到多个数据流。NTFS数据流的表述方式为“文件名:流名”，例如“text.txt:extrastream”。

NTFS流从Windows NT 3.1开始被引入，起初设计目的是为了Services for Macintosh（SFM）能够正确存储Macintosh的资源分岔。现在的Windows服务器已经不再包含此功能，但很多第三方的Apple归档服务（AFP）产品（例如Group Logic的ExtremeZ-IP）仍然会继续使用可选数据流。Internet Explorer和其它一些浏览器会在从网络上下载的文件中添加一个非常小的可选数据流，用于标记它们来自于外部网站（表示可能会存在安全风险），用户在打开这些文件前系统将会显示一个确认提示。[12]当用户表示不希望再次看到这个确认对话框的时候，这个可选流将会从下载的文件中被直接删除。

可选数据流不会显示在Windows资源管理器中，也不会算入查看文件属性时显示的文件大小。如果将文件复制到FAT格式的磁盘、附加到电子邮件、上传到网站，或者移动到任何其它不支持可选流的位置上时，则只有主数据流会被保留下来，其它可选流将被全部丢弃，因此使用可选流来保存重要数据很可能会发生意外。Microsoft提供了一个叫作Streams[13]的工具，用户可以使用这个工具查看卷中的可选流。从Windows Powershell 3.0版本开始，以下cmdlet支持对可选数据流进行操作：Add-Content、Clear-Content、Get-Item、Out-String、Remove-Item，以及Set-Item。[14]

有些媒体播放器也尝试使用可选数据流记录多媒体文件的自定义元数据，用于管理媒体文件。MPEG、OGG等格式通常在文件内签入信息标签记录媒体信息，但不是所有格式都支持这种设计，而使用可选数据流的好处正是他不会影响文件本身的内容。在Windows中注册外壳扩展程序后，系统就可以解析这些数据，然后可以在Windows资源管理器的信息栏中显示它们。但大部分媒体播放器还是选择使用独立数据库而非可选数据流来保存这些信息，因为可选数据流可能带来一些其它问题，一个典型问题是文件上的信息对于所有用户都可见并且是共享的，因此使用可选数据流将无法根据不同用户的安全设置和喜好进行分别管理和保护。

一些恶意软件会使用可选数据流来隐藏程序代码，[15]不过不少恶意软件扫描程序和特殊工具现在已经能够检查可选数据流中的内容。

文件压缩

NTFS能够使用LZNT1算法（LZ77算法的一种变种）压缩文件。[16]文件压缩以16个簇为一个区块进行，也即如果簇大小为4KB，则压缩时单个区块的大小为64KB。NTFS压缩算法支持的最大簇大小为 4KB，如果簇大小超过 4KB，则压缩功能将不可用。[17] 如果压缩可以将64KB数据压缩到60KB或者更小，则NTFS会将多余的4KB页面视为稀疏文件簇，认为他们未经写入。对此类簇的随机访问的性能是可以接受的，操作系统只需跟踪碎片链接即可。但如果处理大型可压缩文件，则会产生大量碎片，因为NTFS会将每个小于64KB的区块都看成一个碎片区域。[18][19]微软NTFS开发团队的研究表明，在簇大小为4KB（默认设置）时，NTFS卷上压缩文件的合理最大尺寸应当在50-60GB之间。当簇大小更小时，最大尺寸也会减小。[20]硬盘空间受限的单用户系统可以使用NTFS压缩在处理小文件（4KB到64KB，或者更大尺寸，具体范围取决于压缩比）时受益。小于900字节的文件将被直接存储在MFT的目录项中。[21]

闪存设备（如固态硬盘）没有传统硬盘的磁头移动延迟，因此对此类设备，磁盘碎片的影响非常有限。具有快速多处理器系统的用户可以通过压缩应用程序文件和数据以提升速度并降低磁盘空间使用率。[22]请注意，使用Sandforce控制器的SSD本身也会压缩数据，但文件系统压缩会导致传输的数据量变少，因此I/O负载会降低。

数据压缩的最佳目标是内容具有重复性、很少写入、通常顺序访问，并且尚未被压缩过的文件。例如，日志文件就是一种理想的压缩目标。

压缩系统引导时需要使用的文件，如驱动程序、NTLDR、winload.exe，或者BOOTMGR，会导致系统无法正确启动。[23]不过在较新版本的Windows系统中，重要的系统文件会被禁止压缩。

当在驱动器或目录的“高级设置”中更改“将文件进行压缩”的设置时，每个文件将被独立进行压缩或者解压缩。

对于压缩文件的读写绝大部分时候是透明的[24]，但Microsoft建议避免对服务器系统或者通过网络共享的远程配置文件进行压缩，因为这可能增加处理器的负担。[25]Microsoft建议不要压缩超过30MB的文件，因为这可能会产生性能问题。由于压缩文件会产生很多碎片，因此磁盘碎片整理过程通常需要花费更长时间。

计算机系统中最慢的设备通常不是CPU而是硬盘，因此NTFS压缩通常可以更有效地利用慢速的非RAM存储系统，节省空间和时间（前提是假设压缩文件的碎片不会连续存放）。[26]

稀疏文件

稀疏文件是包含稀疏数据集的文件，稀疏文件只储存文件中各个有意义的片段，而片段之间的空白将被忽略，这种设计特别适合实际数据非常少、大部分区域空白的文件。读取文件时，对任何被忽略的位置，文件系统程序都会返回数据0，因此文件内容看起来几乎全是零。很多数据库和科学程序使用稀疏文件[27]。Microsoft实现对稀疏文件的高效存储支持，允许应用程序指定文件的空（零）数据区域。读取稀疏文件的应用程序不需要做单独处理，可以继续使用常规方法读取数据，操作系统将根据读取的位置决定返回零或者实际数据。和压缩文件类似，磁盘限额对稀疏文件的尺寸判断以声明大小而非实际占用大小为准。[28][29]

卷影复制

卷影复制（VSC）服务通过将新改写的数据复制到卷影（写入时复制）来保存NTFS卷上的文件和文件夹的历史版本。当用户请求恢复文件的早期版本时，旧的文件数据将会覆盖新数据。该功能也使得数据备份程序可以存档当前系统正在使用的文件。对于负载较重的系统，Microsoft建议将卷影副本设置到单独的磁盘上，以减小系统主要卷的I/O负载。

Windows Vista通过持久卷影副本实现系统还原和先前的版本功能。但旧版本的操作系统加载NTFS卷时，由于其无法识别持久卷影副本的数据格式，这些副本将被删除。

事务

在Windows Vista中，应用程序可以使用事务NTFS（Transactional NTFS，TxF）将一系列对文件的更改归组到一个事务中。事务能够确保所有更改要么同时生效，要么同时作废，并能确保在事务提交完成前，其它应用程序无法无法检测到其中的更改。[30]

该技术使用和卷影复制类似的技术（也即写入时复制）以确保在事务不成功时，被改写的数据可以安全地回滚。通用日志文件系统的日志将记录下尚未成功提交或者已经提交但尚未完全生效的事务（常见原因是事务的某个参与者在提交过程中系统意外崩溃）。

事务NTFS并不要求事务是本机NTFS卷的文件操作，可以包含在其它位置的任意事务数据或操作，例如对其它卷、本地注册表、SQL数据库中、系统服务或者远程服务中的事务修改。所有这些事务使用Windows系统中的“分布事务协调器（DTC）”服务在网络级别协调所有参与者，以确保所有参与者都能接收到同样的提交状态，并传输所有经过确认的更改。分布式NTFS事务的一个典型例子是可以以事务方式创建一个网络级别的分布式文件系统，并且每个客户端都保留每个文件的准确的脱机缓存。

安全

在NTFS中，每个文件或文件夹具有一个安全描述符，用于说明其所有者，并包含两个安全控制列表（ACL）。

第一个列表被称为自主访问控制列表（DACL），用于描述是否允许或禁止特定的用户或用户组进行特定的操作（如读取、写入、执行或删除）。例如，“C:\Program Files”文件夹可能被设定为允许所有用户读取并执行，但只有具有管理员权限的用户才能修改其内容。Windows Vista为DACL增加了强制访问控制功能。DACL是Windows Vista及后续操作系统的用户账户控制功能的主要检查点。

第二个列表被称为系统访问控制列表（SACL），用于描述对文件或文件夹的特定行为是否应当被审核，以及在操作成功后是否应当记录操作。例如，企业可能会对高度敏感的文件开启审核功能，这样管理员就可以了解到是否有人尝试删除或复制这些文件，以及他们的操作是否成功完成。

加密

加密文件系统（EFS）提供对NTFS卷上任意文件和文件夹的用户透明的强保护。加密文件系统需要与EFS服务、Microsoft的加密应用程序接口（CryptoAPI）以及EFS 文件运行时库（FSRTL）联合工作。

EFS使用对称密钥（也被称为“文件加密密钥（FEK）”）加密文件，这比起使用非对称密钥加密在加密和解密大量数据时消耗的时间较少。该对称密钥使用一个和请求加密文件的用户相关的公钥加密文件的内容，加密后的数据储存在被加密文件的可选数据流中。当需要解密文件时，文件系统使用用户的密钥解密储存在文件头中的对称密钥，然后使用该对称密钥解密文件。这些操作在文件系统级别完成，因此对用户来说是透明的。[31]为了处理用户丢失密钥的情况，加密文件系统中还支持多个附加解密密钥，因此除用户外，授权过的恢复代理也能访问数据。NTFS提供的加密和压缩功能是互相排斥的——如果同时希望加密和压缩，则NTFS文件系统级别只能打开其中一种功能，另一种功能需要使用其它第三方工具完成。

Basic、Home和MediaCenter版本的Windows上不支持EFS功能。要使用这个功能，必须安装Professional、Ultimate或者服务器版本的Windows，或者使用Windows域中的企业部署工具进行部署。

限额

磁盘限额是NTFS v3开始提供的功能。此功能允许计算机管理员在受支持的Windows操作系统上为每个用户分别设定允许使用的磁盘空间阈值，或者跟踪查看每个用户使用的磁盘空间使用量。管理员可以设定当某个用户使用特定量的磁盘空间后收到“磁盘空间超限”的警告，甚至拒绝他们继续占用更多空间。如果有文件或者目录使用NTFS文件压缩，则磁盘限额管理的尺寸以压缩后的实际尺寸为准。如果应用程序通过操作系统接口查询用户可用的剩余磁盘空间大小，开启限额后程序得到的大小将是在限额范围内的剩余空间，而不再是磁盘的总剩余空间。

Basic、Home和MediaCenter版本的Windows不支持磁盘限额功能。要使用这个功能，必须安装Professional、Ultimate或者服务器版本的Windows，或者使用Windows域中的企业部署工具进行部署。

重解析点

该功能从NTFS v3开始可用。该功能可以在用户空间中为文件或目录添加一个关联的重解析点属性。当对象管理器（请参见Windows NT线执行）解析文件系统名称并遇到重解析点属性时，它将“重新解析”名称，具体做法是：Windows会将需要重解析的名称传递给已经加载的所有文件过滤驱动程序，每个过滤驱动程序都会检查重解析数据并判断自己是否和该重解析点相关联。如果某个过滤驱动程序判定自己匹配该重解析点，则它将拦截这次文件系统调用，然后执行对应的特定功能。重解析点是实现卷加载点、目录连接、分层存储管理、本机结构存储，以及单实例存储等功能的基础。

卷加载点

类似于Unix 加载点，可以将一个卷的根目录附加到另一个文件系统的某个目录下。这项功能可以让驱动器不需要单独的卷标（如C:或D:）就可以被访问。

当卷被加载到另一个卷的某个目录时，该目录原来的内容将无法访问，而被新加载的卷中的内容所代替。被加载的卷仍然可以继续拥有独立的卷标。NTFS文件系统不允许卷之间相互加载。卷加载点可以是永久的，也可以是非永久的。前者在系统重启后会自动加载，而后者需要手动重新加载。

被加载的卷可以使用NTFS外的其它文件系统。一个常见的例子是，被加载的卷一个远程共享的目录，该目录拥有自己的权限设置，并且能够根据实际文件系统的策略为当前操作系统设定特定的访问权限。

目录连接

类似于卷加载点，但目录连接的连接目标是文件系统中的某个其它目录。例如，目录C:\exampledir带有一个目录连接属性，连接到D:\linkeddir。当用户级别的应用程序访问时，NTFS将自动把所有引用重定向到D:\linkeddir。[1]目录连接功能在概念上类似于Unix的目录符号链接，但符号链接可以连接到任何目标，而NTFS目录连接只允许连接到目录。

目录连接可以在控制台中通过命令MKLINK /J连接名目标目录创建，使用RMDIR连接名删除。目录连接是永久性的，如果在客户端访问服务器的目录连接，则操作系统会使用被链接的目录所属的系统（或域）的安全设置。但连接本身可能拥有独立的安全设置，并且删除一个目录连接不会同时删除目标目录。

有些目录连接是Windows Vista系统创建的，用于保持和早期版本的Windows的兼容性，例如系统驱动器中的Documents and Settings文件夹会被连接到同一个卷中的Users物理目录上。这些目录默认情况下是隐藏的，并且进行相关的安全设置，因此Windows资源管理器不允许外壳或者大部分应用程序直接打开它们，这样的设置可能是为了防止用户发现两个看上去相同的文件夹，然后错误地删除其中的某一个。默认情况下只有本机的SYSTEM账户或者的Administrators用户组成员可以访问这些目录，这是考虑到这些账户安装软件的权限，而安装时可能会产生兼容性问题。

目录连接属于软链接（即使目标目录已经被删除，他们也仍然存在），使用一种类似符号链接的方式工作（只不过对于目标位置和类型有额外的限制），但NTFS文件系统对它们进行特殊优化，解析更快，相比于之后提出的NTFS 符号链接，目录连接的开销更小，且可以在服务器端解析，因此可以在远程共享目录中使用它们。

符号链接

符号链接（或称软链接）从Windows Vista开始引入。[32]符号链接在客户端解析，因此如果服务器上共享一个符号链接，则客户端访问时将服从本机而非服务器端的访问限制。

符号链接可以链接到文件（使用MKLINK符号链接目标文件名创建），也可以链接到目录（使用MKLINK /D符号链接目标目录创建）。和Unix符号链接不同的一点是，NTFS符号链接在创建的时候就要决定目标类型（目录或文件），但创建符号链接的时候并不需要目标已经存在或可以访问，在访问时才会实际检查可访问性。NTFS在访问符号链接时也会检查目标的类型，由于NTFS不允许在任何位置有目录和文件同名，因此如果目标名称存在但是类型不正确，系统也会返回一个找不到目标的错误。

符号链接也可以引用远程主机上的共享文件夹或者其中的文件、子文件夹。但目标并不会被立即加载，而是在应用程序使用OpenFile()或者CreateFile() API请求打开目标的时候才加载到系统中。符号链接是永久的（重新启动后仍然保留在对应的卷上），可以在命令行或者脚本中使用DEL符号链接删除它们。

分层存储管理（HSM）

分层存储管理是一种转移一定时间不用的文件到价值更低的储存介质中的方法。当文件再次被访问时，文件上的重解析点将判定文件需要被使用，并将文件从储存介质中恢复出来。分层存储不但可以节省存储开销，也可以提高操作系统的数据读写和运行效率。

本机结构存储（NSS）

本机结构存储是一种已经被Microsoft终止使用的ActiveX文档存储技术。这项技术允许ActiveX文档使用和ActiveX内部的多流格式相同的方式进行储存。操作系统会加载一个NSS文件过滤驱动程序，可以在应用程序使用文件时透明地处理多流格式。当NSS文件被传输到非NTFS格式的磁盘卷上时，文件内部的多个流将被合并转换为一个流。[33]

分布链接跟踪（DLT）

分布链接跟踪功能允许应用程序跟踪被重命名或者移动到同一计算机、域或工作组的其它卷中的文件、快捷方式和OLE链接。[34]跟踪功能由一个系统服务提供，使用存储在元文件中的对象标识符（OID）索引实现。[35]当应用程序请求跟踪某个文件或目录后，跟踪服务将会创建对象的OID项并指向目标。在一个NTFS v3上执行文件重命名、复制或移动操作时，也会同时复制对象的OID，这样跟踪服务就可以有效地寻找到目标。

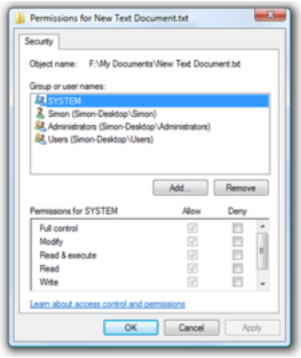
单实例存储（SIS）

当若干个不同目录中存有内容相同的文件时，单实例存储允许将相同文件归并到一个单一文件中，并将所有文件引用到实际的文件上。单实例存储功能包含一个用于管理复制、修改和归并文件的文件系统过滤器和一个用于搜索需要归并的相同文件的用户空间服务（“groveler”）。单实例存储提出时的主要针对目标是远程安装服务器，这些服务器上往往拥有若干个包含大量相同文件的安装镜像，单实例存储可以将它们统一起来，因而节省需要占用的总尺寸。和硬链接不同的一点是，在SIS下，每个文件在逻辑上仍然是独立的，更改被合并为单个文件的任意其中一个文件都不会影响其它文件，而是会取消对该文件的合并并产生一个新的副本。不过新文件不会立即写入到硬盘，NTFS使用类似于写入时复制的技术，在文件最终需要保存时才执行复制。[36]

内部实现

在内部，NTFS使用B+树索引文件系统数据。这种数据结构的方式实现比较复杂，但能够在大多数情况下提高文件的查找速度。文件系统日志用于确保文件的元数据完整，不存在孤立的文件内容。相比于FAT文件系统，NTFS文件系统的可靠性更高。[37]

NTFS允许对名称（包括文件名称、流名称、索引名称等）使用除了0x0000以外的任意16位值序列进行编码。这意味着支持NTFS支持UTF-16码位，但文件系统并不会检查某个UTF-16序列是否有效（也即允许NTFS内部任意16位整数序列，不受Unicode标准的限制）。



Windows Vista系统上的NTFS文件权限设置对话框。

分区引导扇区

NTFS 引导扇区内容

字节偏移	字段长度（字节）	典型值	字段名	描述
0x00	3	0xEB5290	JMP 指令	用于命令处理器继续执行引导扇区后的内容
0x03	8	"NTFS　　" 单词“NTFS”，以及后续四个空格（0x20）	OEM 标识	用于标记当前分区使用NTFS文件系统
0x0B	2	0x0200	每扇区字节数	说明每个扇区的字节数
0x0D	1	0x08	每簇扇区数	说明每个簇包含的扇区数
0x0E	2	0x0000	保留扇区数	被操作系统保留的扇区数，参考文献中未说明保留扇区的位置
0x10	3	0x000000	未使用	该字段的内容始终为零
0x13	2	0x0000	NTFS 未使用	该字段的内容始终为零
0x15	1	0xF8	介质描述符	参考文件中未说明详细信息
0x16	2	0x0000	未使用 Unused	该字段的内容始终为零
0x18	2	0x003F	每磁道扇区数	说明每个磁道的扇区数
0x1A	2	0x00FF	磁头数	该驱动器包含的读写磁头数
0x1C	4	0x0000003F	隐藏扇区数	隐藏的扇区数目。参考文献中未说明隐藏扇区位置
0x20	4	0x00000000	未使用	NTFS未使用
0x24	4	0x80008000	未使用	NTFS未使用
0x28	8	0x00000000007FF54A	总扇区数	该分区包含的总扇区数目
0x30	8	0x0000000000000004	$MFT 簇编号	$MFT 所在的簇的编号
0x38	8	0x0000000000007FF54	$MFTMirr 簇编号	$MFT镜像（备份）所在的簇的编号
0x40	1	0xF6	文件记录段字节数	每个文件记录段的包含的字节数目。如果该值为负数，则实际值为 2 的 -value 次幂，例如值 0xF6 表示记录长度为 2^10（10==-0xf6）
0x41	3	0x000000	未使用	NTFS未使用该字段
0x44	1	0x01	索引缓冲簇数	每个索引缓冲占据的簇数目，算法和文件记录段相同
0x45	3	0x000000	未使用	NTFS未使用该字段
0x48	8	0x1C741BC9741BA514	卷序列数	该分区的唯一随机编号，用于确保内容一致性

字节偏移	字段长度（字节）	典型值	字段名	描述
0x50	4	0x00000000	校验和	表中上述所有项目的校验和。文献中未说明校验算法
0x54	426		启动指令码	用于加载操作系统其它部分的指令码，该字段位置正是引导扇区前三个字节指向的位置
0x01FE	2	0xAA55	扇区结束标记	该字段用于标记一个正常扇区的结束

[38]

操作系统首先通过0x30位置的8个字节找到 $MFT 所在的簇编号，然后将其和每簇扇区数（0x0D位置的1字节）以及每扇区字节数（0x0B位置的2字节）相乘，获得$MFT的字节偏移量。

主文件表（MFT)

在NTFS中，所有文件数据——文件名、创建日期、访问权限（使用访问控制列表（ACL）实现），以及内容——都作为元数据储存在主文件表中。这种抽象的实现方式能够大大简化为文件系统添加功能的成本。例如，Active Directory（活动目录）服务可以很容易在文件系统中为文件添加索引字段。这种设计方式也使得Everything或者Ultrasearch[39]一类的软件可以不依赖于Windows Search实现对文件和文件夹名称的实时搜索。

MFT结构支持最小化磁盘碎片的算法。[40]一个目录项同时包含“文件名”和“文件ID”，后者是用于在主文件表中标识文件的记录编号。文件ID也包含“重用次数”信息，可用于检测对文件的过期引用。这点设计非常类似于Files-11文件系统的W_FID，和NTFS的其他部分迥然不同。

元文件

NTFS包含若干用于定义和组织文件系统的文件。总体来说，这些文件中的绝大多数结构和其它用户文件类似（只有“$Volume”比较特殊），但不能被文件系统客户端直接访问。这些元文件为定义文件、备份文件系统的关键数据、缓存文件系统的更改、管理空闲空间的分配、满足BIOS的要求、跟踪坏扇区单元，以及储存安全信息和磁盘空间使用情况等等多种不同需求提供支持。

区块编号	文件名	作用
0	$MFT	描述卷上的所有文件，包括文件名、时间戳、流名称和数据流所在的簇的编号列表、索引、安全标识符，以及文件属性（如“只读”、“压缩”、“加密”等）。
1	$MFTMirr	$MFT的最开始的几个关键项的副本，通常是4项（4KiB）。
2	$LogFile	文件系统更改的事务日志，用于保护元数据的稳定性。
3	$Volume	卷的相关信息，如卷对象标识符、卷标、文件系统版本，以及若干卷标志（包括：“正在加载”、“需要扫描”、“需要调整$LogFile大小”、“在NT4上加载”、“正在更新卷序列号”、“需要升级结构”等）。这些数据不直接在数据流中存储，而是存储于特殊的 MFT 属性中。如果卷对象标识符存在，则将会在一个 $OJBECT_ID 记录中；卷标存储在 $VOLUME_NAME 记录中；其它数据存储在 $VOLUME_INFORMATION 记录中；卷序列号储存在$Boot文件中（请参见下文）。
4	$AttrDef	所有NTFS项目使用到的属性的定义表，包含属性名称、属性编号和属性描述等。
5	.	根目录。目录数据存储在两个名称均为 $I30 的 $INDEX_ROOT 和 $INDEX_ALLOCATIOn 属性中
6	$Bitmap	一个位图，每一位按顺序指示卷上的对应簇正在被使用（1）或空闲（0）。
7	$Boot	卷引导记录，该文件始终位于卷的第一个簇，其中包含引导代码（用于定位并启动NTLDR/BOOTMGR）、BIOS参数块（其中包含卷序列号），以及$MFT和$MFTMirr所在的簇编号。
8	$BadClus	包含所有标记为“有坏扇区”的簇的一个文件。该文件通常被chkdsk（磁盘扫描）工具使用，用于管理所有簇状态，记录坏扇区，以及标记空白簇。该文件包含两个数据流（无论卷是否有坏道）：一个包含所有坏扇区编号的无名称流（如果卷没有坏扇区则该流长度为零），以及名称为$Bad的流，包含所有包含记录在第一个流中的簇。
9	$Secure	访问控制列表（ACL）数据库，NTFS将所有ACL信息集中存储于该数据库（而非每个文件独立存储各自的ACL）以节省磁盘占用并提高执行效率。此部分包含两个索引，分别是：“$SII”——可能是安全ID索引，以及“$SDH”——安全描述符哈希。注意大部分ACL列表通常都很长，因此这个部分只是索引，记录的是实际存储ACL数据的$SDS流的位置。[1]
10	$UpCase	Unicode大写字母表，用于确保在Win32和DOS名字空间下大小写不敏感。
11	$Extend	文件系统目录，用于包含若干操作系统或其它软件所需要的扩展数据，如$Quota、$ObjId、$Reparse、$UsnJrnl等。
12…23	保留为 $MFT 扩展项使用。扩展项目是一系列附加MFT记录，用于描述未在主记录中包含的属性。例如，当卷需要磁盘清理，包含太多流，具有长文件名或者非常复杂的安全设定，以及其它特殊情况时，可能会用到保留项目	
24	$Extend\$Quota	磁盘限额设置。包含两个索引根目录，名称分别为$O和$Q。
25	$Extend\$ObjId	分布链接跟踪数据。包含一个索引根目录，名称为$O。
26	$Extend\$Reparse	卷上所有重解析点（如符号链接）数据。包含一个索引根目录，名称为$R。
27…	file.ext	常规文件系统项目的起始位置。

Windows对这些元文件的处理方式较为特殊，直接由NTFS.SYS进行处理，因此难以直接查看，需要使用特殊工具进行提取。从Windows 7开始，NTFS驱动程序完全阻挡了用户访问权限，任何尝试访问元文件的请求都会直接进入蓝屏死机界面。微软“OEM支持工具”中的“nfi.exe（NTFS文件扇区信息实用工具）”是一个可查看这些文件的工具l.liru，要查看“$MFT”的内容，只需使用下列命令行：nfi.exe c:\$MFT。另

一个绕过操作系统保护限制的方法是使用7-Zip文件管理器工具并输入低级NTFS路径\\.x:\，此时将会出现三个新的文件夹：$extend、$[DELETED]以及[SYSTEM]。这个操作可以用于任何可移动设备，但如果需要访问当前活动分区，则需要进入离线模式（也即WinRE）。

从MFT到属性、属性表和流

对于每个MFT记录所描述的文件或目录，都有一个线性存放的流描述符（也即属性）存储区，被打包后存放在一个变长记录（也即属性表）中，然后使用额外的填充符填充以满足MFT记录的1Kib对齐要求。这部分数据完整地描述了和文件相关联的所有数据流。注意此处的“流”和文件数据流不是一个概念，而是所有数据信息的统称。

每个流（或称属性）本身包含如下数据：类型（内部通常存储为一个固定长度的整数或者一个描述符，但通常开发应用程序开发时调用FileOpen()或者FileCreate() API时会使用等效的标识符来代表它）、流名称（可选，注意和文件名没有任何关系），以及数据（可选，但大部分流具有数据）。对于NTFS而言，文件的主数据（也即文件内容）、目录的索引信息、文件的可选数据流、以及文件的所有属性，处理方式都是完全相同的，他们都是属性表中的某个属性而已。

- 对于每个MFT中描述的文件（或非常驻的流描述符存储区，此概念请参考下文），每种流描述符（使用流类型+流名称联合区分）必须唯一。此外，NTFS还对于描述符有一些用于排序的约束要求。
- NTFS预定义了一种称为“空流”的类型，用于在流存储区中表明流描述符的结尾。空流必须是每个存储区中的最后一个流描述符，所有之后的剩余空间都将被忽略，只包含填充字节（用于满足MFT的记录尺寸要求或者非常驻流存储区的簇尺寸要求）。
- 对于每个MFT记录，某些类型的流必须存在，例外是整个记录只有一个空流表示该记录未使用。
    - 这里的“某些”具体指的是包含时间戳和其它单比特（Bit）的属性的“标准属性”，这种属性的长度和格式是固定的，这是为了兼容DOS或者Windows 95/98管理的FAT/FAT32文件。
- 某些类型的流不能包含名称，必须是匿名的。
    - 这里的“某些”包括标准属性、NTFS的“文件名”流，以及“短文件名”流（注意如果该文件无需考虑对DOS应用程序的兼容性，则“短文件名”流并不一定出现）。有些时候文件也可能只包含“短文件名”而没有“文件名”，这时候短文件名会代替文件名显示在资源管理器的“文件名”一栏中。
    - 即使流存储区中包含有“文件名”流，也并不表示文件一定会出现在文件系统中。要显示文件，该项目还必须要进入卷上的某个目录的索引表。注意目录的索引表中除了记录该文件的MFT记录编号外，也对文件单独提供一份MFT记录、安全描述符、属性表等，因此理论上单个文件可以被多个目录“硬链接”多次，并且在每个目录中显示为不同的名称。
- 表示文件数据的默认数据流的流类型为$DATA，该数据流没有名称。可选数据流使用同样的流类型，但必须具有流名称。
- 目录的默认数据流和文件的情况不同，目录的数据流（存放文件的索引表）不是匿名的，NTFS会通过目录的数据流的名称（如在NTFS 3以上版本中为“$I30”）来表明目录索引的格式。

可以使用nfi.exe（NTFS文件扇区信息实用工具）查看任意文件或者目录的数据流，这个实用工具目前包含在Microsoft OEM Support Tools中，可以免费发布。[41]

常驻文件和非常驻文件

为了优化小数据文件的资源占用并降低I/O负荷，在流描述符和文件数据大小总和不超过单个MFT最大记录大小时，NTFS会直接将数据放入流描述符区域中（通常情况下，MFT中数据流的内容是实际文件数据锁在的簇的列表）。此类直接将数据存入MFT的文件被称计算机取证工作者称为“常驻数据”。常驻文件允许的数据量受到文件其它信息占用的流描述符大小的影响，通常一个没有可选数据流、文件名不长且无特殊ACL设置的文件允许存放700到800个字节。

- 某些流描述符（如上文提到的首选文件名、基本文件属性，以及下文所述的非常驻流主分配表）必须以常驻形式存在。
- 启用NTFS加密的流、稀疏数据流，以及压缩数据流无法常驻。
- 非常驻流的分配图格式和文件是否为稀疏文件有关。对当前版本的NTFS而言，如果某个非常驻数据流被转换为稀疏流，将无法还原为非稀疏格式，因此该文件将无法重新常驻，除非将流完全截断并且丢弃整个稀疏分配图。
- 如果一个非常驻数据流碎片太多，以至于在MFT记录中无法放下整个分配图，则分配图本身可能也会被存放到一个非常驻流当中，而文件本身只包含一个小型常驻数据流，其内容是标识非常驻分配图所在的簇的位置。
- 如果一个文件的流太多（包括可选数据流、扩展属性、安全描述符等），流描述符无法完整放入MFT中，则NTFS也会创建一个非常驻数据流，用于存放必须常驻的流之外的其它流描述，该数据流使用和MFT一样的格式，但不再有空间尺寸限制。

由于常驻文件不直接占据簇（也即基本的“分配单元”），这使得NTFS卷有可能包含比簇数目更多的文件。例如，一个80GB（74.5GiB）的分区，NTFS可以将其格式化并产生19,543,064个4KiB的簇。除去系统文件（64MiB日志文件，一个2,442,888字节的位图，以及大约25个簇的固定头部），还剩余19,526,158个簇可用于文件和索引。由于每个簇有4个MFT记录，因此卷理论上可以包含将近4×19,526,158 = 78,104,632个常驻文件。

机会锁

机会锁（Oplock）允许网络客户端改变对文件或数据流的缓存策略，以便于增强性能或降低网络占用。[42]机会锁应用到文件某个打开的流上，不影响同一个文件的其它流。

机会锁可以用于在后台透明访问文件。如果没有其它进程访问服务器文件，网络客户端可以避免向文件写入数据；而如果没有其他进程正在写入数据，客户端可以缓存即将读取的数据。

Windows支持四种不同类型的机会锁：

- 等级2（共享）：多个读取，不可写入（也即读缓存）。
- 等级1（独占）：任意缓冲方式的独占访问（也即读写缓存）。
- 批量锁（也是独占的）：在服务器上打开流，在客户端关闭流（读、写和句柄缓存）。
- 过滤锁（也是独占的）：当其它程序尝试访问流时，应用程序和系统过滤器可以放弃该锁（读写缓存，从Windows 2000开始受支持）

在Windows 7和Windows Server 2008 R2系统中，机会锁得到增强，支持每个客户端使用独立的机会锁键。[43]

时间

Windows NT和后续产品使用UTC作为内部时间戳，并在显示时进行适当的转换。也就是说，NTFS时间戳使用UTC模式。

由于历史原因，不支持NTFS的Windows使用本地时区作为时间戳，而当前版本的Windows对非NTFS分区也是用此方式进行处理。这意味着当文件在NTFS和非NTFS分区间进行移动时，操作系统需要实时转换时间戳。如果某些文件移动时处于夏令时，而某些文件处于标准时间，则可能移动后的时间可能会产生歧义，特别是当执行时间正好在时制转换前后时，用户可能会看到一小时的时间差。基于不同地区的不同夏令时规则，在任意12个月中，可能会产生最多4个小时的永久时间戳错误。[44]

互操作性

NTFS具体的内部实现细节属于商业秘密，这给第三方开发者制作NTFS文件系统处理程序带来很大的困难。

Microsoft Windows

尽管绝大多数NTFS版本的绝大部分都完全向前以及向后兼容，但在旧版本的Microsoft Windows加载新版本的NTFS卷仍然会产生不少技术问题。这种问题往往来自于同一台计算机的多重启动功能，或者使用移动硬盘设备传输文件。

例如，在不支持的操作系统上尝试使用带有“先前版本”（严格的说称为卷影副本）的NTFS分区，会导致先前版本数据丢失。[45]

Windows提供了一个命令行工具“convert.exe”可用于将部分文件系统转换为NTFS，包括HPFS（仅在Windows NT 3.1、3.5和3.51中受支持）、FAT16和FAT32（在Windows 2000及后续版本中受支持）。

Mac OS X

Mac OS X 10.3及后续版本包含对NTFS格式分区的只读支持。基于GPL授权的NTFS-3G也可以通过用户空间文件系统在Mac OS X上使用并读写NTFS分区。NTFS-3G的开发团队还提供一个性能更好的商业版本，名称为“Tuxera NTFS for Mac”。[46] Paragon Software Group也出售可执行读写操作的驱动程序，名称为“NTFS for Mac OS X”，[47]部分希捷（Seagate）硬盘包含该组件。[48]Mac OS X 10.6和后续版本中包含有本机NTFS的写入支持，默认情况下此功能未激活，可以通过特定方法打开。但有用户报告此功能不稳定并会导致内核错误，可能这也是该功能未启动或者被宣告的原因之一。[49][50]

Linux

完整并安全的对NTFS的读写功能由NTFS-3G 驱动程序提供。该驱动程序包含在绝大多数Linux发行版中。同时也存在过时的，大部分仅只读的解决方案：

- Linux内核2.2：从版本2.2.0开始，可以读取NTFS分区。
- Linux内核2.6：包含一个由Anton Altaparmakov（来自剑桥大学）和Richard Russon编写的驱动程序，该驱动程序支持读取文件以及在部分情况下的改写文件和调整文件大小。
- NTFSMount：使用ntfsmount可以通过一个用户级驱动程序对文件和目录进行有限的读写操作。[51]

- Tuxera NTFS：高性能可读写商业内核驱动，主要是针对嵌入式设备，他还开发了开源的NTFS-3G驱动。
- NTFS for Linux：由Paragon Software Group提供的对NTFS提供完整读写支持的商用驱动程序，桌面版可免费下载，嵌入式Linux版本则需要收费。
- Captive NTFS（已停止维护）：一个使用Windows自身的驱动程序“ntfs.sys”并进行简单封装的驱动程序。

请注意，上面所有三个用户级别驱动程序（NTFSMount、NTFS-3G以及Captive NTFS）都基于用户空间文件系统（FUSE）实现的，该系统是一个用于在用户空间和内核代码间通讯以获取或保存数据的Linux内核模块。上面所有的驱动程序（除了Tuxera NTFS和Paragon NTFS for Linux）都是以GPL的方式开源的。由于NTFS内部结构非常复杂，内置的2.6.14内核驱动程序和FUSE都不允许修改被认为是不安全的卷，以避免发生数据丢失。

其他

eComStation和FreeBSD都提供对NTFS的只读访问支持（eComStation有一个测试版本的驱动程序允许写入／删除操作，但通常认为该驱动程序还不安全）。BeOS有一个基于NTFS-3G的的第三方工具，允许完整的NTFS读写操作。除了Linux，NTFS-3G也能工作在Mac OS X、FreeBSD、NetBSD、Solaris以及Haiku上。同时，也有一个称为“NTFS4DOS”的商用驱动程序允许在DOS下进行读写。[52] Ahead Software曾经在2002至2004年间开发了一个名为“NTFSREAD”的驱动程序（版本1.200），可用于DR-DOS，并曾包含在他们的Nero Burning ROM软件中。OpenBSD在2011年5月1日发布的4.9版本中提供了针对i386和amd64体系结构的NTFS只读支持。[53] Google开发的Chrome OS操作系统也支持NTFS[54]。

调整大小

有许多第三方工具可以安全地重新调整NTFS分区的大小。在Windows Vista中，Microsoft添加收缩和扩展分区的功能，但该功能非常有限，因为该功能无法整理页面文件碎片或者标记为不可移动的文件，因此限制对分区的收缩能力。取消页面文件重新启动或使用第三方的工具进行磁盘碎片整理也许能改善收缩效果。

另请参阅

- 文件系统的对比
- NTFSDOS
- Files-11—ODS-2非常类似于NTFS（如相比较于INDEXF.SYS和$Mft，BITMAP.SYS和 $Bitmap）
- HPFS，为OS/2操作系统开发的文件系统
- ntfsresize
- Samba（软件）

参考文献

1. Mark Russinovich. Inside Win2K NTFS, Part 1. Microsoft Developer Network. [2008-04-18]. （原始内容存档于2008-04-13）.
2. Microsoft Corporation. How NTFS Works. [2008-01-27]. （原始内容存档于2011-08-24）.

3. Richard Russon and Yuval Fledel. NTFS Documentation. [2007-07-01]. （原始内容存档于2006-02-13）.
4. Custer, Helen. Inside the Windows NT File System. Microsoft Press. 1994. ISBN 978-1-55615-660-1.
5. Kozierok, Charles M. Overview and History of NTFS. PCGuide. April 17, 2001 [2008-10-31]. （原始内容存档于2010-01-02）.
6. Recovering Windows NT After a Boot Failure on an NTFS Drive. Microsoft. November 1, 2006 [2008-10-31]. （原始内容存档于2012-10-24）.
7. New Capabilities and Features of the NTFS 3.1. Microsoft. December 1, 2007 [2011-04-01]. （原始内容存档于2010-09-14）.
8. Change Journals (Windows). MSDN. [2010-04-16]. （原始内容存档于2011-01-17）.
9. Creating, Modifying, and Deleting a Change Journal (Windows). MSDN. [2010-04-16].
0. MS Windows NT Workstation 4.0 Resource Guide, "POSIX Compatibility (http://www.microsoft.com/technet/archive/ntwrkstn/reskit/poscomp.mspx?mfr=true) （页面存档备份 (https://web.archive.org/web/20080501042054/http://www.microsoft.com/technet/archive/ntwrkstn/reskit/poscomp.mspx?mfr=true)，存于互联网档案馆）"
11. MSDN - CreateHardLink function. [14 January 2016]. （原始内容存档于2016-09-10）.
12. Russinovich, Mark E.; Solomon, David A.; Ionescu, Alex. File Systems. Windows Internals 5th. Microsoft Press. 2009: 921. ISBN 978-0-7356-2530-3. “一个使用多数据流的Windows组件例子是附件执行服务[...]，根据文件所下载位置的所属区域不同[...]，Windows资源管理器可能会警告用户”
13. Sysinternals Streams v1.56. [2011-04-01]. （原始内容存档于2011-03-18）.
14. FileSystem Provider. Microsoft. 9 August 2012 [23 January 2015]. （原始内容存档于2015年1月23日）.
15. Malware utilising Alternate Data Streams? (https://www.auscert.org.au/render.html?it=7967) （页面存档备份 (https://web.archive.org/web/20080723180729/https://www.auscert.org.au/render.html?it=7967)，存于互联网档案馆），AusCERT Web Log, 21 August 2007
16. File Compression and Decompression. MSDN Platform SDK: File Systems. [2005-08-18]. （原始内容存档于2019-10-18）.
17. The Default Cluster Size for the NTFS and FAT File Systems. Microsoft. January 31, 2002 [2012-01-10]. （原始内容存档于2012-01-30）.
18. Understanding NTFS Compression. [2011-03-16]. （原始内容存档于2011-06-29）.
19. Shrinking the gap: carving NTFS-compressed files. [2011-05-29]. （原始内容存档于2011-10-07）.
20. Middleton, Dennis. Understanding NTFS Compression. Ntdebugging Blog. Microsoft. 20 May 2008 [2011-04-01]. （原始内容存档于2011-06-29）.
21. How NTFS Works. 2003-03-28 [2011-10-24]. （原始内容存档于2016-07-30）.
22. Should You Compress Data On Your SSD?. 2011-12-01 [2013-04-05].
23. Disk Concepts and Troubleshooting. Microsoft. [2012-03-26]. （原始内容存档于2012-04-20）.
24. Read-Only Filegroups and Compression. SQL Server 2008 Books Online (November 2009). [2010-04-20]. （原始内容存档于2010-03-06）.
25. "Best practices for NTFS compression in Windows (http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251186) （页面存档备份 (https://web.archive.org/web/20131113184821/http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251186)，存于互联网档案馆）." Microsoft Knowledge Base. Retrieved on 2005-08-18.

26. Daily, Sean. Optimizing Disks. IDG books. January 1998 [2007-12-17]. （原始内容存档于2007-12-17）.
27. 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) （页面存档备份 (https://web.archive.org/web/20100516122013/http://blogs.msdn.com/psssql/archive/2009/03/04/sparse-file-errors-1450-or-665-due-to-file-fragmentation-fixes-and-workarounds.aspx)，存于互联网档案馆），Microsoft Customer Service and Support (CSS) SQL Server Engineers Blog, March 4, 2009
28. Sparse Files. MSDN Platform SDK: File Systems. [2005-05-22]. （原始内容存档于2019-10-18）.
29. Sparse FIles and Disk Quotas. Win32 and COM Development: File Systems. [2007-12-05]. （原始内容存档于2008-02-24）.
30. Transactional NTFS. MSDN. [2007-02-02]. （原始内容存档于2008-10-11）.
31. How EFS Works (http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distrib/dsck_efs_duwf.mspx) （页面存档备份 (https://web.archive.org/web/20070427062206/http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distrib/dsck_efs_duwf.mspx)，存于互联网档案馆），Microsoft Windows 2000 Resource Kit
32. Symbolic Links (Windows). MSDN. [2017-05-28]. （原始内容存档于2016-02-06）.
33. John Saville, "What is Native Structured Storage? (http://www.windowsitpro.com/Article/ArticleID/13785/13785.html) （页面存档备份 (https://web.archive.org/web/20070927212324/http://www.windowsitpro.com/Article/ArticleID/13785/13785.html)，存于互联网档案馆）"
34. 存档副本. [2013-07-25]. （原始内容存档于2016-03-12）.
35. http://technet.microsoft.com/en-us/library/cc736811（WS.10）.aspx
36. Single Instance Storage in Windows 2000 (PDF). Microsoft Research and Balder Technology Group. [2008-10-31]. （原始内容存档 (PDF)于2008-10-29）.
37. "Microsoft TechNet Resource Kit (http://www.microsoft.com/technet/archive/ntwrkstn/reskit/file_sys.mspx?mfr=true) （页面存档备份 (https://web.archive.org/web/20080224040805/http://www.microsoft.com/technet/archive/ntwrkstn/reskit/file_sys.mspx?mfr=true)，存于互联网档案馆）"
38. NTFS Partition Boot Sector (http://www.ntfs.com/ntfs-partition-boot-sector.htm) （页面存档备份 (https://web.archive.org/web/20160719125827/http://www.ntfs.com/ntfs-partition-boot-sector.htm)，存于互联网档案馆) Information on structure of the boot sector.
39. UltraSearch v1.7 – Freeware for Ultra-Fast File Search. [2013-10-17]. （原始内容存档于2013-04-25）.
40. Master File Table. MSDN. July 2, 2012.
41. OEM Support Tools Phase 3 Service Release 2 Availability. Microsoft Corporation. 2007-02-21 [2010-06-16]. （原始内容存档于2015-02-23）. “Windows NT File System (NTFS) File Sector Information Utility [...] A tool used to dump information about an NTFS volume”
42. 存档副本. [2013-07-25]. （原始内容存档于2010-08-23）.
43. http://technet.microsoft.com/en-us/library/ff383236（WS.10）.aspx

14. "Beating the Daylight Saving Time bug and getting correct file modification times (http://www.codeproject.com/datetime/dstbugs.asp)（页面存档备份 (https://web.archive.org/web/20041114032240/http://www.codeproject.com/datetime/dstbugs.asp)，存于互联网档案馆）" The Code Project
15. cfsbloggers. How restore points and other recovery features in Windows Vista are affected when dual-booting with Windows XP. The Filing Cabinet. July 14, 2006 [2007-03-21].（原始内容存档于2006-07-18）.
16. Tuxera NTFS for Mac. Tuxera. August 30, 2011 [September 20, 2011].（原始内容存档于2013-10-17）.
47. NTFS for Mac OS X, communication channel between Mac OS X and Windows. Paragon Software Group. [September 20, 2011].（原始内容存档于2013-10-17）.
48. Seagate Read/Write NTFS driver for Mac OS X. [2013-10-17].（原始内容存档于2011-02-10）.
49. Alvares, Milind. Snow Leopard's hidden NTFS read/write support. 2009-10-02 [18 September 2010].（原始内容存档于2010年8月10日）.
50. How to Write to NTFS Drives on a Mac. [2018-04-05].（原始内容存档于2018-04-05）.
51. "ntfsmount wiki page on linux-ntfs.org (http://wiki.linux-ntfs.org/doku.php?id=ntfsmount)（页面存档备份 (https://web.archive.org/web/20070927213220/http://wiki.linux-ntfs.org/doku.php?id=ntfsmount)，存于互联网档案馆）"
52. Recovery information. [2008-10-31].（原始内容存档于2008-11-09）.
53. 存档副本. [2013-10-17].（原始内容存档于2013-10-17）.
54. Chromebook 支持的文件类型和外接设备 (https://support.google.com/chromebook/answer/183093)

- Bolosky, William J.; Corbin, Scott; Goebel, David; & Douceur, John R. Single Instance Storage in Windows 2000 (PDF). Microsoft Research & Balder Technology Group, Inc. date [2008-10-31].（原始内容存档 (PDF)于2008-10-29）.
- Custer, Helen. Inside the Windows NT File System. Microsoft Press. 1994. ISBN 978-1-55615-660-1.
- Nagar, Rajeev. Windows NT File System Internals: A Developer's Guide. O'Reilly. 1997. ISBN 978-1-56592-249-5.

外部链接

- 文献：
  - Low-level description of NTFS disk structures (http://sourceforge.net/project/showfiles.php?group_id=13956&package_id=16543)（页面存档备份 (https://web.archive.org/web/20090201080954/http://sourceforge.net/project/showfiles.php?group_id=13956&package_id=16543)，存于互联网档案馆） from the Linux-NTFS project
  - NTFS.com (http://www.ntfs.com/)（页面存档备份 (https://web.archive.org/web/20100107030918/http://www.ntfs.com/)，存于互联网档案馆） – documentation and resources for NTFS
  - Microsoft NTFS Technical Reference (https://web.archive.org/web/20080704051237/http://technet2.microsoft.com/WindowsServer/en/library/81cc8a8a-bd32-4786-a849-03245

d68d8e41033.mspx)

- 实现技术：
  - NTFS-3G (http://www.ntfs-3g.org/)（页面存档备份 (https://web.archive.org/web/20090917181509/http://www.ntfs-3g.org/)，存于互联网档案馆） – an open source read/write NTFS driver for Linux, FreeBSD, Mac OS X, NetBSD, Solaris and Haiku.
  - Linux-NTFS (http://linux-ntfs.org/)（页面存档备份 (https://web.archive.org/web/20090106004433/http://linux-ntfs.org/)，存于互联网档案馆） – an open source project to add NTFS support to the Linux kernel（write support is limited, but can be used for simple tasks），and write POSIX-compatible utilities for accessing and manipulating NTFS（ntfsprogs; includes ntfsls, ntfsresize, ntfsclone, etc）. Linux NTFS FAQ (https://web.archive.org/web/20081112060709/http://www.linux-faqs.com/faq/misc/ntfs.php) and howto (https://web.archive.org/web/20081207193302/http://www.linux-faqs.com/Forum/viewtopic.php?t=7)
  - Captive NTFS (http://www.jankratochvil.net/project/captive/)（页面存档备份 (https://web.archive.org/web/20100114190757/http://www.jankratochvil.net/project/captive/)，存于互联网档案馆） – a shim which used the Windows NTFS driver to access NTFS file systems under Linux

---

取自 “https://zh.wikipedia.org/w/index.php?title=NTFS&oldid=66679734”

本页面最后修订于2021年7月20日 (星期二) 12:01。

本站的全部文字在知识共享 署名-相同方式共享 3.0协议之条款下提供，附加条款亦可能应用。（请参阅使用条款）
Wikipedia®和维基百科标志是维基媒体基金会的注册商标；维基™是维基媒体基金会的商标。
维基媒体基金会是按美国国内税收法501(c)(3)登记的非营利慈善机构。