**NTFS文件系统研究**

bzMe.Work

NTFS提供FAT中没有的性能，可靠性和高级功能。例如，NTFS包括内置安全功能（如文件和文件夹权限）和加密文件系统（EFS），后者是用于在NTFS卷上存储加密文件的技术。NTFS还支持最大256TB的卷，支持磁盘配额和压缩，并支持已安装的驱动器。

**第一部分 NTFS文件构成**

下面是一个NTFS卷（Volume）（或者称为分区）的组织结构，由4部分构成，如图所示：

Organization of an NTFS Volume

**✓** NTFS引导扇区（NTFS Boot Sector）：它包含BIOS参数块，该参数块存储有关卷布局和文件系统结构的信息，以及加载Windows操作系统的引导代码。

**✓**主文件表（Master File Table，MFT）：包含从NTFS分区检索文件所需的信息，例如文件的属性。

**✓**文件系统数据（File System Data）：存储主文件表（MFT）中不包含的数据。

**✓**主文件表拷贝（Master File Table Copy）：它包含文件系统恢复所必需的记录的拷贝（如果原始拷贝有问题）。

**一、NTFS引导扇区（NTFS Boot Sector）**

在MBR磁盘上，位于每个分区的第一个逻辑扇区的引导扇区是启动计算机的关键磁盘结构。它包含可执行代码和代码所需的数据，包括文件系统用于访问卷的信息。在格式化卷时创建引导扇区。引导扇区的末尾是一个2字节的结构，称为签名或扇区标记的末尾，它总是被设置为0x55AA。在运行Windows的计算机上，活动分区上的引导扇区将加载到内存并启动Ntldr（它将CPU切换到受保护模式，并启动文件系统，然后读取Boot.ini文件的内容，此信息确定启动选项和初始启动菜单选择），如果安装了多个版本的Windows, Ntldr将加载引导菜单；如果只安装了一个操作系统，Ntldr将加载操作系统。

引导扇区由以下要素组成：

1、一个基于x86框架的CPU跳转指令**（CPU jump instruction）**。

2、原始设备制造商标识**（OEM ID）**。

3、BIOS参数块**（BPB）**。

4、扩展BPB**（Extended BPB）**。

5、启动操作系统的引导代码**（Bootstrap code）**。

在操作系统启动前，MBR将CPU执行转移到引导扇区，因此引导扇区的前三个字节必须是有效的——基于x86的可执行的**CPU跳转指令**，它跳过接下来的几个不可执行字节。遵循跳转指令的是8字节的**OEM ID**，这是一串字符，用于标识格式化卷的操作系统的名称和版本号，为了保持与MS-DOS的兼容性，Windows在这个字段中记录了“NTFS”。在OEM ID之后是**BPB**，它提供了使可执行引导代码能够定位Ntldr的信息。BPB总是以相同的偏移量开始，因此标准参数位于已知位置。磁盘大小封装在BPB中。因为引导扇区的第一部分是x86跳转指令，所以将来可以通过在末尾添加新信息来**扩展BPB**。跳转指令只需要稍加调整就可以适应这种变化。BPB以打包（未对齐）格式存储。

当格式化NTFS卷时，格式化程序为引导扇区和引导代码分配前16个扇区。下表描述了引导扇区的构成：

|  |  |  |
| --- | --- | --- |
| 字节偏移 | 字段长度（字节） | 字段名称 |
| 0x00 | 3 bytes | 跳转指令**（Jump instruction）** |
| 0x03 | 8 bytes | 制造商标识**（OEM ID）** |
| 0x0B | 25 bytes | BIOS参数块**（BPB）** |
| 0x24 | 48 bytes | 扩展BPB**（Extended BPB）** |
| 0x54 | 426 bytes | 引导代码**（Bootstrap code）** |
| 0x01FE | 2 bytes | 扇区结束标志**（End of sector marker）** |

图示如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | Jump instruction | | | OEM ID | | | | | | | | Bytes/Sector | | Sectors/  cluster | reserved | |
| 10 | 0x000000 | | | unused | | Media desc | 0x0000 | | Sect /track | | Number heads | | Hidden Sectors | | | |
| 20 | unused | | | | | | | | Total Sectors | | | | | | | |
| 30 | Logical Cluster of $MFT | | | | | | | | Logical Cluster of $MFTMirr | | | | | | | |
| 40 | Clust / File record segment | | | | Clusters / Index Block | | | | Volume Serial Number | | | | | | | |
| 50 | Checksum | | | | Bootstrap code | | | | | | | | | | | |
| 60 | Bootstrap code | | | | | | | | | | | | | | | |
| … |
| 1E0 |
| 1F0 | Bootstrap code | | | | | | | | | | | | | | 55 | AA |

NTFS Boot sector参考：

<https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v%3dws.10)>

<http://technet.microsoft.com/en-us/library/cc976796.aspx>

上图各色表示的含义：

|  |
| --- |
| Jump instruction（跳转指令） |
| OEM ID（制造商标识） |
| BPB（BIOS参数块） |
| Extended BPB（扩展的BPB） |
| Bootstrap code（引导代码） |
| End of sector marker（扇区结束标志） |

下面对上图中涉及的术语进行解释：

|  |  |  |  |
| --- | --- | --- | --- |
| 字节偏移 | 字段长度（字节） | 样本值（十六进制） | 字段名称和解释 |
| 0x0B | 2 bytes | 00 02 | **每扇区字节数**。硬盘扇区的大小，美国使用的大多数磁盘，这个字段的值是512。  [**Bytes Per Sector**.The size of a hardware sector. For most disks used in the United States, the value of this field is 512.] |
| 0x0D | 1 byte | 08 | **每簇扇区数**。簇中扇区的数量。  [**Sectors Per Cluster.**The number of sectors in a cluster.] |
| 0x0E | 2 bytes | 00 00 | **保留扇区数**。总是为0，因为NTFS将引导扇区放在分区的开头，如果值不为0，NTFS将无法挂载卷。  [**Reserved Sectors**. Always 0 because NTFS places the boot sector at the beginning of the partition. If the value is not 0, NTFS fails to mount the volume.] |
| 0x10 | 3 bytes | 00 00 00 | 值必须为0，否则NTFS无法挂载卷。  [Value must be 0 or NTFS fails to mount the volume.] |
| 0x13 | 2 bytes | 00 00 | 值必须为0，否则NTFS无法挂载卷。  [Value must be 0 or NTFS fails to mount the volume. ] |
| 0x15 | 1 byte | F8 | **媒体描述符**。提供有关正在使用的媒体的信息。0xF8表示硬盘，0xF0表示高密度3.5英寸软盘。媒体描述符条目是MS-DOS FAT16磁盘的遗产，在Windows中没有使用。  **[Media Descriptor**. Provides information about the media being used. A value of F8 indicates a hard disk and F0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows.] |
| 0x16 | 2 bytes | 00 00 | 值必须为0，否则NTFS无法挂载卷。  [Value must be 0 or NTFS fails to mount the volume.] |
| 0x18 | 2 bytes | 3F 00 | 未使用或由NTFS检查。  [Not used or checked by NTFS.] |
| 0x1A | 2 bytes | FF 00 | 未使用或由NTFS检查。  [Not used or checked by NTFS.] |
| 0x1C | 4 bytes | 3F 00 00 00 | 未使用或由NTFS检查。  [Not used or checked by NTFS.] |
| 0x20 | 4 bytes | 00 00 00 00 | 值必须为0，否则NTFS无法挂载卷。  [Value must be 0 or NTFS fails to mount the volume.] |
| 0x24 | 4 bytes | 80 00 80 00 | 未使用或由NTFS检查。  [Not used or checked by NTFS.] |
| 0x28 | 8 bytes | 1C 91 11 01  00 00 00 00 | **总扇区数**。硬盘上扇区的总数。  [**Total Sectors**. The total number of sectors on the hard disk.] |
| 0x30 | 8 bytes | 00 00 04 00  00 00 00 00 | **$MFT文件的逻辑簇号**。通过使用MFT的逻辑簇号来标识MFT的位置。  [**Logical Cluster Number for the File $MFT**. Identifies the location of the MFT by using its logical cluster number.] |
| 0x38 | 8 bytes | 11 19 11 00  00 00 00 00 | **$MFTMirr文件的逻辑簇号**。通过使用MFT的逻辑簇号标识MFT镜像拷贝的位置。  [**Logical Cluster Number for the File $MFTMirr**. Identifies the location of the mirrored copy of the MFT by using its logical cluster number.] |
| 0x40 | 1 byte | F6 | **每个MFT记录的簇数。**  [**Clusters Per MFT Record**. The size of each record. NTFS creates a file record for each file and a folder record for each folder that is created on an NTFS volume. Files and folders smaller than this size are contained within the MFT. If this number is positive (up to 7F), then it represents clusters per MFT record. If the number is negative (80 to FF), then the size of the file record is 2 raised to the absolute value of this number.] |
| 0x41 | 3 bytes | 00 00 00 | 未被NTFS使用。（Not used by NTFS.） |
| 0x44 | 1 byte | 01 | **每个索引缓冲区的簇数。**  [**Clusters Per Index Buffer**. The size of each index buffer, which is used to allocate space for directories. If this number is positive (up to 7F), then it represents clusters per MFT record. If the number is negative (80 to FF), then the size of the file record is 2 raised to the absolute value of this number.] |
| 0x45 | 3 bytes | 00 00 00 | 未被NTFS使用。  [Not used by NTFS.] |
| 0x48 | 8 bytes | 3A B2 7B 82  CD 7B 82 14 | **卷的序列号。**  [**Volume Serial Number**. The volume’s serial number.] |
| 0x50 | 4 bytes | 00 00 00 00 | 未被NTFS使用。  [Not used by NTFS.] |
|  |  |  |  |

BPB and Extended BPB Fields on NTFS Volumes 参考：

<https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v%3dws.10)>

<http://technet.microsoft.com/en-us/library/cc976796.aspx>

**二、主文件表（Master File Table，MFT）**

当您使用NTFS格式化卷时，Windows将在分区上创建**主文件表**（Master File Table，MFT）和**元数据文件**（Metadata Files）。

MFT是一个关系数据库（relational database），由**文件记录行**（rows of file records）和**文件属性列**（columns of file attributes）组成。

它为NTFS卷上（NTFS volume）的每个文件至少包含一个条目（entry），包括MFT本身。如下图所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **主文件表**（Master File Table，MFT） | | | | | | |
| **文件记录行**（rows of file records） | **文件属性列**（columns of file attributes） | | | | | |
| Standard Information | Attribute List | File Name | Data | Object ID | … |
| $Mft |  |  |  |  |  |  |
| $MftMirr |  |  |  |  |  |  |
| $LogFile |  |  |  |  |  |  |
| … |  |  |  |  |  |  |

**1、MFT和元数据文件（MFT and Metadata Files）-**（文件记录行（rows of file records））

由于MFT存储关于它自己的信息，NTFS将MFT的前16条记录保存为元数据文件(大约16KB)，这些元数据文件用于描述MFT。元数据文件以美元符号($)开头。MFT的其余记录则包含卷上每个文件和文件夹的记录。下面列表描述存储在MFT（主文件表）中的元数据文件：

|  |  |  |  |
| --- | --- | --- | --- |
| 文件 | 名称 | MFT记录号 | 描述 |
| $Mft | Master file table  主文件表 | 0 | 包含NTFS卷上每个文件和文件夹的一个基本文件记录。如果文件或文件夹的分配信息太大，无法放入单个记录中，那么还会分配其他文件记录。  [Contains one base file record for each file and folder on an NTFS volume. If the allocation information for a file or folder is too large to fit within a single record, other file records are allocated as well. ] |
| $MftMirr | Master file table mirror  主文件表镜像 | 1 | 确保在单扇区故障时访问MFT。它是MFT前四条记录的副本镜像。  [Guarantees access to the MFT in case of a single-sector failure. It is a duplicate image of the first four records of the MFT. ] |
| $LogFile | Log File  日志文件 | 2 | 包含NTFS用于更快恢复的信息。Windows使用日志文件在系统失败后将元数据一致性恢复到NTFS。日志文件的大小取决于卷的大小，但是可以使用Chkdsk命令增加日志文件的大小。）（也就是说，日志文件记录了文件的增删改操作，当出现故障导致文件损坏后，系统将根据日志文件进行恢复。  [Contains information used by NTFS for faster recoverability. The log file is used by Windows to restore metadata consistency to NTFS after a system failure. The size of the log file depends on the size of the volume, but you can increase the size of the log file by using the Chkdsk command.] |
| $Volume | Volume  卷标 | 3 | 包含有关卷的信息，如卷标签和卷版本。  [Contains information about the volume, such as the volume label and the volume version. ] |
| $AttrDef | Attribute definitions  属性定义 | 4 | 列出属性名称、数字和描述。  [Lists attribute names, numbers, and descriptions.] |
| . | Root file name index | 5 | 根文件夹。  [The root folder.] |
| $Bitmap | Cluster bitmap | 6 | 通过显示空闲和未使用的簇来表示卷。  [Represents the volume by showing free and unused clusters.] |
| $Boot | Boot sector | 7 | 包括用于挂载卷的BPB，以及在卷可引导时使用的附加引导加载程序代码。  [Includes the BPB used to mount the volume and additional bootstrap loader code used if the volume is bootable.] |
| $BadClus | Bad cluster file | 8 | 包含卷已损坏的簇。  [Contains bad clusters for a volume.] |
| $Secure | Security file | 9 | 包含卷中所有文件的惟一安全描述符。  [Contains unique security descriptors for all files within a volume.] |
| $Upcase | Upcase table | 10 | 将小写字符转换为匹配的Unicode大写字符。  [Converts lowercase characters to matching Unicode uppercase characters.] |
| $Extend | NTFS extension file | 11 | 用于各种可选扩展，如配额、重新解析点数据和对象标识符。  [Used for various optional extensions such as quotas, reparse point data, and object identifiers.] |
|  |  | 12–15 | 保留以备将来使用。  [Reserved for future use.] |
|  |  |  |  |

Metadata Files Stored in the MFT 参考：

<https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v%3dws.10)>

<http://technet.microsoft.com/en-us/library/cc781134(WS.10).aspx>

上表介绍了MFT的文件记录行，下面介绍MFT的文件属性列。

**2、NTFS文件记录属性（NTFS File Record Attributes）-**（文件属性列（columns of file attributes））

NTFS卷上的每个分配扇区都属于一个文件。甚至文件系统元数据也是文件的一部分。NTFS将每个文件(或文件夹)视为一组文件属性。文件元素（如名称、安全信息，甚至数据）都是文件属性。每个属性由一个属性类型代码和一个可选的属性名称标识。

文件和文件夹记录各为1KB存储在MFT中，其属性写入MFT中分配的空间。除了文件属性外，每个文件记录还包含关于文件记录在MFT中的位置的信息。

当一个文件的属性能够适合该文件的MFT文件记录时，它们就称为常驻属性。诸如文件名和时间戳之类的属性总是常驻的。当文件的信息量不符合其MFT文件记录时，一些文件属性就会变为非驻留属性，系统分配一个或多个磁盘空间簇的非驻留属性，文件/文件夹非驻留属性的一部分保留在MFT中，并指向外部簇。

NTFS创建属性列表属性（Attribute List attribute ）来描述所有属性记录的位置。下表列出了当前由NTFS定义的文件属性类型：

|  |  |
| --- | --- |
| 属性类型 | 描述 |
| Standard Information | 访问模式(只读、读/写等)、时间戳和链接计数等信息。  [Information such as access mode (read-only, read/write, and so forth) timestamp, and link count.] |
| Attribute List | 不适合MFT记录的所有属性记录的位置。  [Locations of all attribute records that do not fit in the MFT record.] |
| File Name | 用于长文件名和短文件名的可重复属性。文件的长名称最多可达255个Unicode字符。短名称是8.3，不区分大小写的文件名称。POSIX需要的附加名称或硬链接可以包含为附加文件名属性。  [A repeatable attribute for both long and short file names. The long name of the file can be up to 255 Unicode characters. The short name is the 8.3, case-insensitive name for the file. Additional names, or hard links, required by POSIX can be included as additional file name attributes.] |
| Data | 文件数据。NTFS支持每个文件的多个数据属性。每个文件通常有一个未命名的数据属性。一个文件还可以有一个或多个命名数据属性。  [File data. NTFS supports multiple data attributes per file. Each file typically has one unnamed data attribute. A file can also have one or more named data attributes.] |
| Object ID | 卷唯一的文件标识符。由分布式链接跟踪服务使用。不是所有的文件都有对象标识符。  [A volume-unique file identifier. Used by the distributed link tracking service. Not all files have object identifiers.] |
| Logged Tool Stream | 类似于数据流，但是操作被记录到NTFS日志文件中，就像NTFS元数据更改一样。EFS使用此属性。  [Similar to a data stream, but operations are logged to the NTFS log file just like NTFS metadata changes. This attribute is used by EFS.] |
| Reparse Point | 用于安装驱动器。可安装文件系统(IFS)筛选驱动程序也使用这种方法将某些文件标记为该驱动程序的特殊文件。  [Used for mounted drives. This is also used by Installable File System (IFS) filter drivers to mark certain files as special to that driver.] |
| Index Root | 用于实现文件夹和其他索引。  [Used to implement folders and other indexes.] |
| Index Allocation | 用于实现大文件夹和其他大索引的B树结构。  [Used to implement the B-tree structure for large folders and other large indexes.] |
| Bitmap | 用于实现大文件夹和其他大索引的B树结构  [Used to implement the B-tree structure for large folders and other large indexes.] |
| Volume Information | 仅在$Volume系统文件中使用。包含卷版本。  [Used only in the $Volume system file. Contains the volume version.] |
|  |  |

**NTFS File Attribute Types 参考：**

<https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v%3dws.10)#ntfs-file-record-attributes>

<http://technet.microsoft.com/en-us/library/cc976808.aspx>

NTFS为每个文件创建一个文件记录，为每个文件夹创建一个文件夹记录。MFT包含一个单独的MFT文件记录。

这些文件和文件夹记录各为1KB，并存储在MFT中。文件的属性被写入MFT中分配的空间中。如下所示：

MFT Entry with Resident Record

下面列出一些重要的文件属性类型的结构：

**$Standard\_Information**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | A | | B | | | C | D | E | | F | |
| 0 | Date Created\* | | | | | | | | Date Modified | | | | | | | | | | | | | |
| 10 | Date MFT record modified | | | | | | | | Date Accessed | | | | | | | | | | | | | |
| 20 | Flags | | | | Max Versions | | | | Version Num | | | | | | | Class ID | | | | | | |
| 30 | Owner ID | | | | Security ID | | | | Quota Charged | | | | | | | | | | | | | |
| 40 | Update Sequence Number | | | | | | | |  | |  | |  | |  | |  |  | |  | |  |

\*Time values are in 100 nanoseconds since January 1, 1601 UTC。

Flags (used for both $Standard\_Information and $File\_Name)：

|  |  |  |
| --- | --- | --- |
| Bit | Hex | Meaning |
| 0 | 0x0001 | Read only |
| 1 | 0x0002 | Hidden |
| 2 | 0x0004 | System |
| 3 | 0x0008 |  |
| 4 | 0x0010 |  |
| 5 | 0x0020 | Archive |
| 6 | 0x0040 | Device |
| 7 | 0x0080 | Normal |
| 8 | 0x0100 | Temporary |
| 9 | 0x0200 | Sparse File |
| A | 0x0400 | Reparse Point |
| B | 0x0800 | Compressed |
| C | 0x1000 | Offline |
| D | 0x2000 | Not Indexed |
| E | 0x4000 | Encrypted |
| F | 0x8000 |  |

参考：<http://msdn.microsoft.com/en-us/library/aa365535(v=VS.85).aspx>

**$File\_Name**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | Parent Directory | | | | | | | | | Date Created | | | | | | | |
| 10 | Date Modified | | | | | | | | | Date MFT Modified | | | | | | | |
| 20 | Date Accessed | | | | | | | | | Logical file size | | | | | | | |
| 30 | Size on disk | | | | | | | | | Flags\* | | | | Reparse value | | | |
| 40 | Name len | len Name type | | Name (variable length) | | | | | | | | | | | | | |

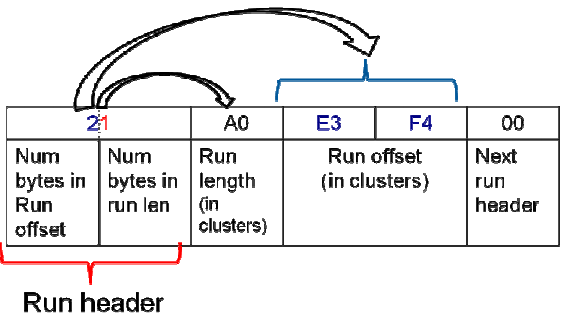
Name types

|  |  |
| --- | --- |
| Value | Description |
| 0 | POSIX (unicode, case sensitive) |
| 1 | Win32 (unicode, case insensitive) |
| 2 | DOS (8.3 ASCII, case insensitive) |
| 3 | Win32 7 DOS (when Win32 fits in DOS space) |

**$Data** (Standard Header with data run, may be resident or non resident, non resident shown here)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | | 2 | 3 | | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | Type ID 0x80 | | | | | | Attribute Length | | | | Form code | name len | Name offset | | flags | | Atrib ID | |
| 10 | Start virtual cluster number | | | | | | | | | | Ending virtual cluster number | | | | | | | |
| 20 | Runlist offset | | Compression unit size | | | 0x0000 | | | | | Size of attribute content | | | | | | | |
| 30 | size on disk of attribute content | | | | | | | | | | Initialized size of attribute content | | | | | | | |
| 40 | Data runlists | | | | | | | | | | | | | | | | | |

Data runlists



**$ATTRIBUTE\_LIST entry**（one entry per attribute in the record, including attributes that precede the list）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | | 8 | 9 | A | | B | C | D | E | F |
| 0 | Type ID | | | | Record Length | | | Atrib name len | | Attrib name offset | Lowest VCN | | | | | | | | |
| 10 | $MFT Record number | | | | | | Seq num | | | | Reserved | | | Start of name (if present) | | | | | |

参考：http://msdn.microsoft.com/en-us/library/bb470038%28v=vs.85%29.aspx

除了文件属性外，每个文件记录还包含关于文件记录在MFT中的位置的信息。带有常驻记录的MFT条目显示了一个小文件或文件夹的MFT记录的内容。小文件和文件夹(通常为900字节或更小)完全包含在文件的MFT记录中。

通常，每个文件使用一个文件记录。然而，如果一个文件具有大量属性或变得高度碎片化，它可能需要多个文件记录。如果是这种情况，文件的第一条记录，即基本文件记录，则存储文件所需的其他文件记录的位置。

当NTFS枚举大文件夹中的文件时，使用B树结构的好处是显而易见的。B树结构允许NTFS对类似的文件名进行分组或索引，然后只搜索包含该文件的组，从而最小化查找特定文件所需的磁盘访问次数，特别是对于大型文件夹。由于B树结构，NTFS对于大文件夹的性能优于FAT，因为FAT必须在列出所有文件之前扫描大文件夹中的所有文件名。

**小结：**前面说过，MFT是一个关系数据库（relational database），由**文件记录行**（rows of file records）和**文件属性列**（columns of file attributes）组成。通过上面的介绍，可以看到NTFS的文件组织结构大概是这样的：



系统通过读写不同的元数据表进行不同的操作，例如：对文件的增删改操作都记录在$LogFile元数据表中，以便于数据出现故障后能及时恢复；如果在写入期间发生错误，NTFS将包含坏扇区的簇地址放在$BadClus元数据表中，这样坏扇区就不会被重用。

下面对一些重要的元数据表进行分析。

**第二部分 NTFS元数据表分析**

NTFS中包含两种日志文件：

**✓**一种是事务日志文件（Transaction Log），记录在**$LogFile**元数据表中；

**✓**一种是变更日志文件（Change Log），记录在**$UsnJrnl**元数据表中。

**一、日志记录表（$LogFile）**

**（一）事务日志介绍（About Transaction Log）**

NTFS是一个可恢复的文件系统，它通过使用标准的事务日志（transaction log）和恢复技术（recovery techniques）来保证卷的一致性。在发生系统故障时，NTFS运行一个恢复过程，该过程访问存储在事务日志文件中的信息。NTFS恢复过程保证将卷恢复到一致的状态。事务日志只需要很少的开销。

NTFS将修改卷上文件的每个操作视为事务，并将每个操作作为一个整体单元来管理。NTFS还可能将一个复杂的操作分解为多个事务。事务启动后，要么完成它，要么发生导致操作失败的事件，然后回滚它，NTFS卷返回到事务开始前的状态。可能导致操作失败的事件包括坏扇区（bad sectors,）、暂态低内存条件（transient low-memory conditions）和断开连接的设备（disconnected devices.）。

为确保事务可以完成或回滚，NTFS对每个事务执行以下步骤：

1、将事务的元数据操作记录在内存中的缓存日志文件中。

2、在内存中记录实际的元数据操作（metadata operation）。

3、将缓存日志文件中的事务标记为已提交。

4、将日志文件刷新到磁盘。

5、将实际的元数据操作刷新到磁盘。

步骤4和5在事务完成后以延迟的方式发生，这意味着刷新操作不绑定到事务本身。相反，NTFS在内存中快速修改日志和元数据，然后在方便的时候刷新，以提高性能。

NTFS保证在将事务中修改的元数据写入磁盘之前，将包含事务元数据操作的日志记录写入磁盘。在NTFS更新缓存之后，NTFS通过在缓存的日志文件中记录事务已完成来提交事务。将缓存的日志文件刷新到磁盘后，所有提交的事务都将保证完成，即使在将更改写入磁盘之前系统发生故障。

如果发生系统故障，NTFS日志中有足够的信息来完成或中止任何部分NTFS事务。在恢复操作期间，NTFS将重做日志文件中找到的每个提交的事务。然后NTFS在日志文件中定位系统故障时未提交的事务，并撤消日志文件中记录的每个元数据操作。因为NTFS在将任何元数据更改写入磁盘之前将日志刷新到磁盘，所以NTFS拥有关于在恢复期间需要回滚的任何元数据更改（any metadata changes）的完整信息。

**（二）日志记录表介绍（About $LogFile）**

在系统中，$LogFile元数据表即是NTFS的事务日志文件（transaction log file）。

如果因电源故障或关键系统故障导致系统意外停机，操作系统会将文件系统状态恢复到以前的状态，并将信息保存在$LogFile文件中。

下面是与$LogFile相关的关键要素：

1、$LogFile中包含文件的所有事务操作记录，其中包括：

**✓**文件或目录的创建。

**✓**文件或目录的删除。

**✓**对$data的修改。

**✓**对MFT条目（MFT entry）的修改。

2、每条$LogFile记录都有LSN（$LogFile序列号）。

3、每条$LogFile记录都有用于恢复的操作数据（operation data）和操作前数据（data before operation）。通过操作数据，我们可以进行重新操作（Redo）；通过操作前数据，我们可以进行撤销操作（Undo）。

4、每个卷中都有一个$LogFile文件，以记录该卷中文件的增删改等操作。

5、$LogFile位于MFT的2号条目。如下示：

|  |  |  |
| --- | --- | --- |
| Entry Number | File Name | Stored Information |
| 0 | $Mft | MFT Entry |
| 1 | $MftMirr | Backup of $MFT |
| 2 | $LogFile | Transaction Log |
| 3 | $Volume | Volume label, Identifier, Version |
| … | … | … |

**（三）日志记录表的大小（Size of $LogFile）**

**✓**一般$LogFile的大小为64 MB。

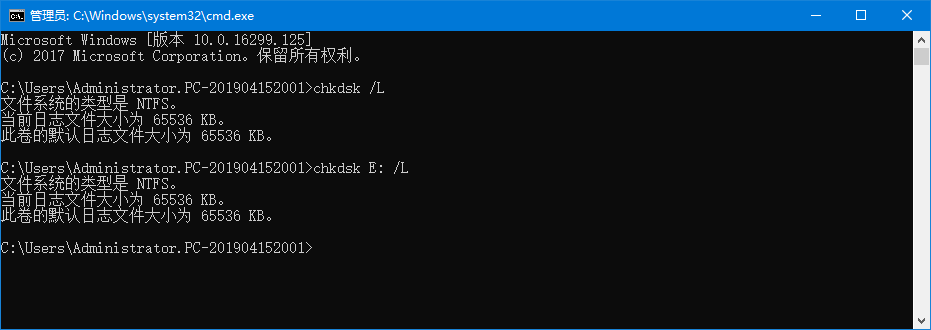
**✓**大小可以根据卷大小进行更改，但通常小于等于64 MB，且只能递增不能减少。

**✓**在典型的电脑使用情况下（如上网，处理文件等），容量64 MB可以在$LogFile记录中保存2~3个小时的活动。

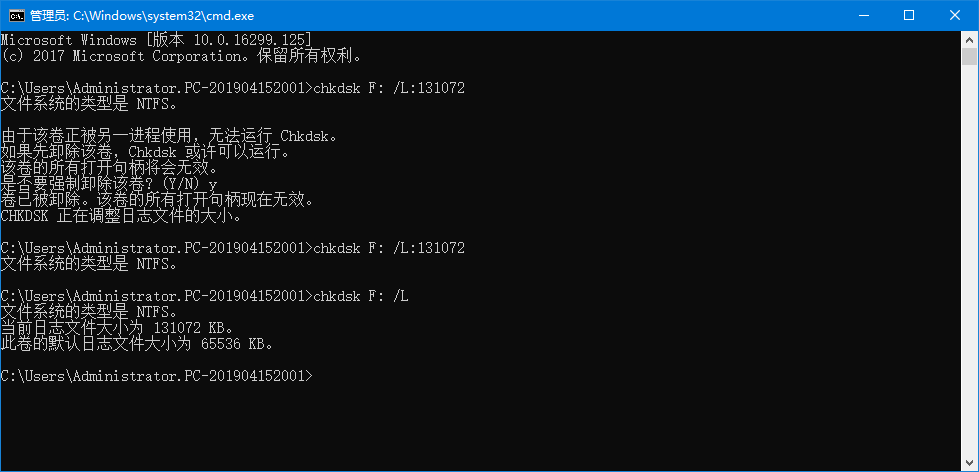
**✓**为了便于取证，应该增加$LogFile的大小。

**（四）调整日志记录表的大小（Resize of $LogFile）**

可以在cmd中，通过chkdsk /L查看当前卷标的$LogFile大小，或者chkdsk X: /L查看指定卷标的$LogFile大小，如下所示：



还可以通过“/L : [filesize(KB)]”命令修改$LogFile的大小，但当前卷正在使用中，是不允许修改$LogFile的大小的，必须先卸载卷才能修改，如下所示：



卸载使用中的卷，调整$LogFile的大小后，大小不再是默认的64MB，而是128MB了。

**（五）日志记录表的结构（The Structure of $LogFile）**

$LogFile的结构分成两个区域，分别是：重启区域（Restart Area）和日志记录区域（Logging Area）。

每个区域以页面（Page）作为基本单位，一个页面的大小是0x1000字节（4MB）。

重启区域（Restart Area）：

**✓**该区域记录上次操作的信息，也称为当前操作的记录；

**✓**重启区域的位置位于$LogFile中的第一个和第二个页面(0x0000~0x2000)。

日志记录区域（Logging Area）：

**✓**该区域记录实际操作记录；

**✓**它位于重启区域（Restart Area）(0x2000~)之后；

**✓**它分为“缓存页面区域（Buffer Page Area）”和“正常页面区域（Normal Page Area）”。

如下图所示：



**1、重启区域（Restart Area）**

**✓**重启区域记录最近或当前操作记录的信息，其中“Current LSN”包含上次操作记录的LSN信息。

**✓**重启区域连续两页，第二页为备份页，且每个页面都以魔术数值（Magic Number）“RSTR”开头。

**✓**重启区域的结构如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | A | | B | C | | D | E | | F |
| 0 | **“RSTR”** **(Magic Number)** | | | | Update Sequence Offset | | Update Sequence Count | | | Check Disk LSN | | | | | | | | | | |
| 10 | System Page Size | | | | Log Page Size | | | | | Restart Offset | | | Minor Version | | | Major Version | | | Update Sequence Array | |
| 20 | Update Sequence Array | | | | | | | | | | | | | | | | | | | |
| 30 | **Current LSN** | | | | | | | | | Log Client | | | Client List | | | Flags | | | | |

**2、日志记录区域（Logging Area）**

**✓**日志记录区域记录实际的操作记录。

**✓**该区域分为“缓存页面区域（Buffer Page Area）”和“正常页面区域（Normal Page Area）”。

**（1）缓存页面区域（Buffer Page Area）**

**✓**缓存页面区域占用日志记录区域的前两页(0x2000~0x4000)，第二页用于备份；

**✓**操作记录按顺序存储；

**✓**如果页面填满记录，则将页面内容移动到“正常页面区域”；

**✓**因此，最后的操作记录总是存储在这个区域。



**（2）正常页面区域（Normal Page Area）**

**✓**日志记录区域的其余部分（缓存页面区域除外）（0x4000~）为正常页面区域；

**✓**操作记录按顺序存储；

**✓**如果区域中充满了记录，则从区域的开始覆盖这些记录。



**3、页面的结构（Logging Area）**

**✓**页面由一个页头（Page Header）和多个操作记录（Operation Records）构成；

**✓**如果最后一个操作记录不能容纳在一个页面中，则存储其余的记录内容在下一个连续页。

**（1）页头的结构（The Structure of Page Header）**

页头存储页面的元数据，其结构如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | | 6 | 7 | 8 | 9 | A | | B | C | | D | E | F |
| 0 | “RCRD” (Magic Number) | | | | Update Sequence | | Offset Update Sequenc | | | Last LSN or File Offset | | | | | | | | | |
| 10 | Flags | | | | Page Count | | Page Position | | | Next Record Offset | | | Word Align | | | DWord Align | | | |
| 20 | Last End LSN | | | | | | | | | Update Sequence Array | | | | | | | | | |
| 30 | Update Sequence Array | | | | | | | | | | | | | | | | | | |

|  |  |
| --- | --- |
| Magic Number | “RCRD”。 |
| Last LSN | 记录中（包括跨页记录）最高LSN。  [the highest LSN among the records including the record of crossed the page.] |
| Next Record Offset | 页面中拥有最高LSN的记录的偏移量。  [the offset of record having the highest LSN in page.] |
| Last End LSN | 除跨页记录外，其他记录中的最高LSN。  [the highest LSN among the records except record that crossed the page] |
|  |  |

**（2）操作记录的结构（The Structure of Operation Records）**

操作记录存储事务操作（transaction operation）的实际内容。

一个事务操作按顺序由多个操作记录（operation records）组成。如下图所示：



这多条记录中包含有3种类型的记录：

**✓**检查点记录（Check Point Record）：事务的开始记录（the start record of transaction）。

**✓**更新记录（Update Record）：事务的中间记录（the middle records of transaction）。

**✓**提交记录（Commit Record）：事务的最后记录（the last record of transaction）。

**✓**除“检查点记录”外，所有操作记录均有上一操作记录（previous operation record）的信息，便于操作记录回滚。

一条操作记录由头部（Header）和数据（Data）构成。

**✓**头部（Header）：它是记录的元数据（the meta data of record），固定大小(0x58)（88个字节）；

**✓**数据（Data）：它由重做（Redo）数据和撤销（Undo）数据组成，其中：

•重做（Redo）：是指操作完成后的数据（例如：写操作（write operation）的写数据（written data））；

•撤销（Undo）：是指操作前的数据（例如：写操作（write operation）开始前（before operation）的数据））。

也就是说一条记录，如果写入成功后，可以再次写入（Redo）；如果写入失败了，可以回滚回原来的数据（Undo）。

当一个事务操作执行时出现错误的时候，操作系统将使用“Commit Record”中的“Previous LSN”来执行回滚操作，并应用“Undo”数据，来恢复到事务执行前的状态。如下所示：



操作记录的结构（The Structure of Operation Record）如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | | B | C | D | E | | F |
| 0 | This LSN | | | | | | | | | Previous LSN | | | | | | |  | | |
| 10 | Client Undo LSN | | | | | | | | | Client Data Length | | | | | Client ID | | | | |
| 20 | Record Type | | | | | Transaction ID | | | | Flags | | | Alignment or Reserved | | | | | | |
| 30 | Redo OP | | | Undo OP | | Redo Offset | | Redo Length | | Undo Offset | | | Undo Length | | Target Attribute | | | LCNs to follows | |
| 40 | Record Offset | | | Attr Offset | | MFT Cluster Index | | Alignment or Reserved | | Target VCN | | | | | Alignment or Reserved | | | | |
| 50 | Target LCN | | | | | Alignment or Reserved | | | |  | | | | | | | | | |

|  |  |
| --- | --- |
| This LSN | 当前记录的LSN。  [LSN of current record.] |
| Previous LSN | 上一记录的LSN。  [LSN of previous record.] |
| Client Undo LSN | 如果发生错误需要恢复时，它记录执行“Undo”操作时的LSN。  [In case of a error recovery, a LSN information of record has following ‘Undo’ operation.] |
| Client Data Length | 记录大小，从“Redo OP”字段开始到记录结束。  [Size of Record(from “Redo Op” field to end of the record)] |
| Record Type | 记录类型，0x02（Check Point Record）， 0x01（Update Record/ Commit Record）。 |
| Flags | 标志，0x01（记录跨当前页），0x00（记录没有跨当前页）。  [0x01(record cross the current page), 0x00(record doesn’t cross the current page).] |
| Redo Op | 重做操作码。  [Redo operation code] |
| Undo Op | 撤销操作码。  [Undo operation code] |
| Redo Offset | 重做数据偏移量（来自“Redo Op”字段）。  [Offset of “Redo” data（from “Redo Op” field）] |
| Redo Length | 重做数据的大小。  [Size of “Redo” data] |
| Undo Offset | 撤销数据偏移量（来自“Undo Op”字段）。  [Offset of “Undo” data（from “Undo Op” field）] |
| Undo Length | 撤销数据的大小。  [Size of “Undo” data] |
| LCNs to Follows | 0x01（还有下一个记录），0x00（没有下一个记录）。  [0x01(There is a next record), 0x00(There is no next record)] |
| Record Offset | 记录偏移量。假如是对MFT记录进行操作，则此值是在MFT记录中应用的重做/撤消数据的属性偏移量；假如是其余的操作，则此值是0x00。  [In case of operation to MFT record, the offset of attribute applied Redo/Undo data within the MFT record;  In case of the rest operation, the value is 0x00.] |
| Attr Offset | 属性偏移量。假如是对MFT记录进行操作，则此值是属性中应用重做/撤消数据的点的偏移量；假如是其他操作，则此值是在簇中应用重做/撤消数据的点的偏移量。  [In case of operation to MFT record, the offset of point applied Redo/ Undo data within the attribute;  In case of other operation, the offset of point applied Redo/Undo data within the cluster.] |
| MFT Cluster Index | MFT簇索引。假如是对MFT记录操作，则此值是在簇中应用重做/撤消数据的记录位置。其中：First (0x0000)，Second(0x0002)，Third (0x0003), forth(0x0006)。  [In case of operation for MFT record, the location of record applied Redo/Undo data within cluster. First (0x0000), Second(0x0002), Third (0x0003), forth(0x0006)] |
| Target VCN | 应用重做/撤消数据的$MFT文件的虚拟簇号。  [VCN(Virtual Cluster Number) of “$MFT” file applied Redo/Undo data.] |
| Target LCN | 应用重做/撤消数据的磁盘的逻辑簇号。  [LCN(Logical Cluster Number) of the disk applied Redo/Undo data.] |
|  |  |

重做/撤销操作码（**Redo/Undo Operation Code**（Redo OP/Undo OP）），如下表所示：

|  |  |
| --- | --- |
| **NTFS Operation** | **Hex Value** |
| Noop | 0x00 |
| CompensationlogRecord | 0x01 |
| InitializeFileRecordSegment | 0x02 |
| DeallocateFileRecordSegment | 0x03 |
| WriteEndofFileRecordSegement | 0x04 |
| CreateAttribute | 0x05 |
| DeleteAttribute | 0x06 |
| UpdateResidentValue | 0x07 |
| UpdataeNonResidentValue | 0x08 |
| UpdateMappingPairs | 0x09 |
| DeleteDirtyClusters | 0x0A |
| SetNewAttributeSizes | 0x0B |
| AddindexEntryRoot | 0x0C |
| DeleteindexEntryRoot | 0x0D |
| AddIndexEntryAllocation | 0x0F |
| SetIndexEntryVenAllocation | 0x12 |
| UpdateFileNameRoot | 0x13 |
| UpdateFileNameAllocation | 0x14 |
| SetBitsInNonresidentBitMap | 0x15 |
| ClearBitsInNonresidentBitMap | 0x16 |
| PrepareTransaction | 0x19 |
| CommitTransaction | 0x1A |
| ForgetTransaction | 0x1B |
| OpenNonresidentAttribute | 0x1C |
| DirtyPageTableDump | 0x1F |
| TransactionTableDump | 0x20 |
| UpdateRecordDataRoot | 0x21 |
|  |  |

**（六）日志记录表的事件分析（The Event Analysis of $LogFile）**

**二、变更记录表（$UsnJrnl）**

**（一）什么是$UsnJrnl？**

**1、$UsnJrnl简介**

-------------------------------------------------------------------------------------------------------------------------------------------------

参考：<https://en.wikipedia.org/wiki/USN_Journal>

The **USN Journal** (**U**pdate **S**equence **N**umber **Journal**), or **Change Journal**,[[1]](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/USN_Journal.html#cite_note-ms-web-msdn-msj-1-1) is a feature of [NTFS](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/NTFS.html) which maintains a record of changes made to the [volume](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Volume_(computing).html). It is not to be confused with the journal used for the [NTFS file system journaling](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/NTFS.html).

USN日志(更新序列号日志)或变更日志是NTFS的一个特性，它维护对卷所做更改的记录。不要将它与用于NTFS文件系统日志记录的日志混淆。

When [Windows 2000](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Windows_2000.html) was released, [Microsoft](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Microsoft.html) created [NTFS](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/NTFS.html) version 3.0, which included several new features and improvements over older versions of the [file system](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/File_system.html). One of these was a new system management feature that is very useful for certain types of [applications](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Application_software.html). Under Windows 2000, NTFS 3.0 [partitions](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Disk_partitioning.html) can be set to keep track of changes to files and directories on the [volume](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Volume_(computing).html), providing a record of when and what was done to the various objects. When enabled, the system records all changes made to the volume in the USN Journal, which is the name also used to describe the feature itself.

当Windows 2000发布时，微软创建了NTFS 3.0版本，其中包含了一些新特性，并对旧版本的文件系统进行了改进。其中之一是一个新的系统管理特性，对于某些类型的应用程序非常有用。在Windows 2000下，可以设置NTFS 3.0分区来跟踪卷上文件和目录的更改，提供对不同对象的操作时间和操作的记录。当启用时，系统将在USN日志中记录对卷所做的所有更改，USN日志也是用来描述特性本身的名称。

One journal is maintained for each NTFS volume and stored in the [NTFS metafile](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/NTFS.html) named $Extend\$UsnJrnl. It begins as an empty [file](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Computer_file.html). Whenever a change is made to the volume, a record is added to the file. Each record is identified by a [64-bit](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/64-bit.html) Update Sequence Number or USN. (For this reason Change Journals are sometimes called USN Journals.) Each record in the Change Journal contains the USN, the name of the file, and information about what the change was.

每个NTFS卷维护一个日志，并存储在名为$Extend\$UsnJrnl的NTFS元文件中。它以一个空文件开始。每当对卷进行更改时，都会向文件添加一条记录。每个记录由64位更新序列号或USN标识。(由于这个原因，变更日志有时被称为USN日志。)变更日志中的每条记录都包含USN、文件的名称以及关于变更内容的信息。

The Change Journal describes the changes that took place using bit flags (e.g. USN\_REASON\_DATA\_OVERWRITE[[2]](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/USN_Journal.html#cite_note-USN_RECORD_V2_structure-2)), therefore it does not include all the [data](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Data.html) or details associated with the change. For this reason the Change Journal cannot be used to undo operations on files within NTFS.

变更日志描述了使用位标志(例如USN\_REASON\_DATA\_OVERWRITE )发生的变更，因此它不包含与变更相关的所有数据或细节。因此，不能使用更改日志撤消NTFS内文件上的操作。

-------------------------------------------------------------------------------------------------------------------------------------------------

变更记录表，即更新序号变更日志元数据表（$UsnJrnl）（**$**:元数据表, **Usn**:Update Sequence Number, **Jrnl**:Change Journal），顾名思义，它是从序号的维度来统计，并确定应用程序在特定文件中发生的任何更改。

在Win7中，默认情况下会激活Journal功能，在Win XP中它是停用的，但可以通过Fsutil工具激活，命令格式如下：

**fsutil usn [createjournal] m=<MaxSize> a=<AllocationDelta> <VolumePath>**

详细参见：<http://technet.microsoft.com/en-us/library/cc788042.aspx>

**2、$UsnJrnl的组成**

$UsnJrnl由**$Max**属性（$Max attribute）和**$J**属性（$J attribute）组成。

**$Max**：

它存储变更日志的元数据。（The meta data of change log is stored.）

**$J**：

它存储实际更改的日志记录。（The actual change log records are stored.）特性如下：

**✓**每条记录都有USN(更新序列号)信息。（Each record has USN(Update Sequence Number) information. ）

**✓**记录顺序由USN决定。（The record order is determined with USN. ）

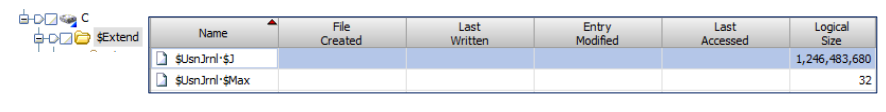
**✓**USN = $J属性中记录的偏移量。（USN = the offset value of a record within $J attribute）

**✓**USN信息也存储在MFT记录的$STANDARD\_INFORMATION属性中。（USN information is also stored in then

$STANDARD\_INFORMATION attribute of a MFT record）

**3、$UsnJrnl位于何处**

$UsnJrnl文件位于$Extend文件夹下。（The file is located under “$Extend” folder.）如下图所示：



可以通过以下工具提取或查看$UsnJrnl文件信息：

EnCase Forensic：法证软件，用于司法鉴定，罪证采集等（<https://www.guidancesoftware.com/>）。

Winhex：默认情况下，此工具按物理大小提取文件。

ExtractUsnJrnl：该工具只能提取除稀疏区域（sparse area）外的有效数据。 ( <https://github.com/jschicht/ExtractUsnJrnl>)

**4、$UsnJrnl的大小**

通常情况下：

如果全天使用(24小时/天)，则记录1~2天的日志。

在常规使用情况下(8小时/天)，记录4~5天的日志。

因此，为了延长日志记录时间，可以创建更大的$UsnJrnl日志大小。

**（二）变更记录表的结构（The Structure of $ UsnJrnl）**

**1、$Max属性的结构（The Structure of $Max attribute）**

**✓**$Max属性的大小占用32个字节的固定大小。

**✓**$Max属性的结构如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Size（Byte） | Stored Information | Detail |
| 0x00 | 8 | Maximum Size | 日志数据的最大大小。  [The maximum size of log data.] |
| 0x08 | 8 | Allocation Size | 保存新日志数据时分配区域的大小。  [The size of allocated area when new log data is saved.] |
| 0x10 | 8 | USN ID | $UsnJrnl文件的创建时间(FILETIME)。  [The creation time of "$UsnJrnl" file(FILETIME)] |
| 0x18 | 8 | Lowest Valid USN | 当前记录中USN的最小值  [The least value of USN in current records] |
|  |  |  |  |

**2、$J属性的结构（The Structure of $J attribute）**

**✓**可变大小的日志记录被连续列出。（The log records of variable size are listed consecutively.）

**✓**零填充的“稀疏区域”占据属性的前端部分。（The zero-filled "Sparse Area" occupies front part of an attribute. ）如下图所示：



✓采用这种结构的原因是操作系统保持$J属性中保存的日志数据的大小相同。（The reason for this structure is because the operating system keeps the same size of the log data saved in the $J attribute.）

✓$J属性的记录分配策略如下（The record allocation policy of $J attribute）

1. 新的日志记录被添加到属性的末尾。（The new log records are added at the end of the attribute.）

2. 如果添加记录的总大小超过“分配大小”，操作系统将确保整个日志数据的大小超过“最大大小”。（If the total size of the added records exceeds "**Allocation Size**", the operation system assures that the size of the entire log data exceeds "**Maximum Size**".）

3.如果整个日志数据的大小超过“最大大小”，则属性的前端区域与“分配大小”大小相同，都为零。（If the size of the entire log data exceeds "**Maximum Size**", the front area of attribute is occupied by zero as much as size of "Allocation Size".）

✓因此，$J属性的逻辑大小不断增长，但是保存实际数据的区域的大小却是保持不变的。（Thus, the logical size of $J attribute grow continuously, but the size of area saving actual data is kept constant.）  
✓日志数据的一般大小是0x200000 ~ 0x23FFFFF(2097152~268435235)

**3、$UsnJrnl的记录结构（The Record Structure of $ UsnJrnl）**

$ UsnJrnl的记录分4个版本，分别是：

USN\_RECORD，USN\_RECORD\_V2，USN\_RECORD\_V3，USN\_RECORD\_V4。

USN RECORD 2.0版结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Size（Byte） | Stored Information | Detail |
| 0x00 | 4 | Size of Record |  |
| 0x04 | 2 | Major Version | 2  本条记录的变更日志的主要版本号。如果变更日志记录的版本是2.0（USN\_RECORD\_V2），那么主要版本号是2；如果变更日志记录的版本是2.0（USN\_RECORD\_V3），那么主要版本号是3；如果变更日志记录的版本是3.0（USN\_RECORD\_V3），那么主要版本号是3，其它以此类推。由于本条记录的版本是2.0，因此此处是2。 |
| 0x06 | 2 | Minor Version | 0  本条记录的变更日志记录的次要版本号。如果变更日志记录是2.0版本，那么次要版本号是0。 |
| 0x08 | 8 | MFT Reference  Number | 受当前更改事件影响的文件或目录的“MFT参考编号”。此值将日志记录与文件关联起来。  ["MFT Reference Number" of file or directory that effected by currently change event. This value associates a journal record with a file.] |
| 0x10 | 8 | Parent MFT  Reference Number | 受当前更改事件影响的文件或目录的父目录的“MFT参考号”。使用此信息和$MFT可以获得完整的路径信息。此值将日志记录与父目录关联起来。  ["MFT Reference Number" of parent directory of file and directory that effected by currently change event. The full path information can be obtained with this information and $MFT. this value associates a journal record with a parent directory.] |
| 0x18 | 8 | USN | 此条记录的USN（更新序列号）。  [The USN of this record(Update Sequence Number).] |
| 0x20 | 8 | TimeStamp(FILETIME) | 事件时间(UTC + 0)。此记录的标准UTC时间戳(FILETIME)，为64位格式。  [Event Time(UTC +0).The standard UTC time stamp ([FILETIME](https://msdn.microsoft.com/9baf8a0e-59e3-4fbd-9616-2ec9161520d1)) of this record, in 64-bit format.] |
| 0x28 | 4 | Reason Flag | 变更事件的标志。重命名或移动操作生成两条USN记录，一条记录该项的旧父目录，另一条记录新父目录。  [The flag of change event. A rename or move operation generates two USN records, one that records the old parent directory for the item, and one that records a new parent.] |
| 0x2C | 4 | Source Information | 触发事件变化的主体。有关更改源的附加信息，由DeviceIoControl操作的FSCTL\_MARK\_HANDLE设置。当一个线程写入一个新的USN记录时，只有当线程也设置了这些标志时，先前记录中的源信息标志才会继续出现。因此，源信息结构允许应用程序过滤掉仅由已知源(例如反病毒过滤器)设置的USN记录。  [The subject that triggers change of event. Additional information about the source of the change, set by the [FSCTL\_MARK\_HANDLE](https://msdn.microsoft.com/c96b49d8-12f3-4281-9f9f-6621769359f0) of  the [DeviceIoControl](https://msdn.microsoft.com/1d35c087-6672-4fc6-baa1-a886dd9d3878) operation.When a thread writes a new USN record, the source information flags in the prior record continues to be present only if the thread also sets those flags. Therefore, the source information structure allows applications to filter out USN records that are set only by a known source, for example, an antivirus filter.] |
| 0x30 | 4 | Security ID | 分配给与此记录关联的文件或目录的唯一安全标识符。  [The unique security identifier assigned to the file or directory associated with this record.] |
| 0x34 | 4 | File Attributes | 受当前事件影响的对象的属性信息。通常，它用于将对象分类到文件或目录中。与此记录关联的文件或目录的属性，由GetFileAttributes函数返回。  [The attribute information of the object effected by current event. Generally, it is used for classifying the object into a file or directory. The attributes for the file or directory associated with this record, as returned by the [GetFileAttributes](https://msdn.microsoft.com/1d35c087-6672-4fc6-baa1-a886dd9d3878) function.] |
| 0x38 | 2 | Size of Filename | 受当前事件影响的对象名称的大小。与此记录关联的文件或目录的名称的长度，以字节为单位。使用此成员来确定文件名长度，而不是依赖于在文件名中以“\0”结尾来分隔文件名。  [The size of object name effected by current event. The length of the name of the file or directory associated with this record, in bytes. Use this member to determine file name length, rather than depending on a trailing '\0' to delimit the file name in FileName.] |
| 0x3A | 2 | Offset to Filename | 记录中对象名称的偏移量。它是从结构开始的文件名成员的偏移量。  [The offset of object name within record. The offset of the FileName member from the beginning of the structure.] |
| 0x3C | N | Filename | 受当前事件影响的对象(文件或目录)名称。  此字段是与此记录关联的文件或目录的Unicode格式名称，这个文件或目录名的长度是可变的。  在处理文件名时，不要指望文件名后面有一个'\0'分隔符，而是使用FileNameLength（Size of Filename）来确定文件名的长度。  不要使用文件名执行任何编译时指针运算。相反，在运行时使用FileNameOffset成员的值进行必要的计算。这样做有助于使您的代码与USN\_RECORD\_V2的任何未来版本兼容。  The object(file or directory) name effected by current event.  The name of the file or directory associated with this record in Unicode format. This file or directory name is of variable length.  When working with **FileName**, do not count on the file name that contains a trailing '\0' delimiter, but instead determine the length of the file name by using **FileNameLength**（Size of Filename）.  Do not perform any compile-time pointer arithmetic using **FileName**. Instead, make necessary calculations at run time by using the value of the **FileNameOffset（**Offset to Filename**）** member. Doing so helps make your code compatible with any future versions of **USN\_RECORD\_V2**. |
|  |  |  |  |

USN\_RECORD\_V2参考来源：

<https://docs.microsoft.com/zh-cn/windows/desktop/api/winioctl/ns-winioctl-usn_record_v2>

<http://msdn.microsoft.com/en-us/library/aa365722.aspx>

|  |  |
| --- | --- |
| Parent MFT Reference Number | 受当前更改事件影响的文件或目录的父目录的“MFT参考号”。使用此信息和$MFT可以获得完整的路径信息。此值将日志记录与父目录关联起来。  ["MFT Reference Number" of parent directory of file and directory that effected by currently change event. The full path information can be obtained with this information and $MFT. this value associates a journal record with a parent directory.]  使用“父MFT参考编号（Parent MFT Reference Number）”代替“MFT参考编号（MFT Reference Number）”的原因是：如果使用“MFT参考编号”，则在删除相关文件时可能无法获得完整的路径信息。  [The reason for using "Parent MFT Reference Number“ instead of "MFT Reference Number"：If "MFT Reference Number“ is used, full path information may not be obtained when relevant file is deleted.] |

**Reason Flag**

|  |  |
| --- | --- |
| Value | Meaning |
| **USN\_REASON\_DATA\_OVERWRITE**  0x00000001 | 文件或目录中的数据已经被覆盖。  [The data in the file or directory is overwritten.] |
| **USN\_REASON\_DATA\_EXTEND**  0x00000002 | 文件或目录被添加到。  [The file or directory was added to] |
| **USN\_REASON\_DATA\_TRUNCATION**  0x00000004 | 文件或目录被截断。  [The file or directory was truncated.] |
| **USN\_REASON\_NAMED\_DATA\_OVERWRITE**  0x00000010 | 文件的一个或多个命名数据流中的数据被覆盖。  [The data in one or more named data streams for a file is overwritten.] |
| **USN\_REASON\_NAMED\_DATA\_EXTEND**  0x00000020 | 文件的一个或多个命名数据流被扩展(添加到)。  [The one or more named data streams for a file are extended (added to).] |
| **USN\_REASON\_NAMED\_DATA\_TRUNCATION**  0x00000040 | 文件的一个或多个命名数据流被截断。  [The one or more named data streams for a file is truncated.] |
| **USN\_REASON\_FILE\_CREATE**  0x00000100 | 第一次创建文件或目录。  [The file or directory was created for the first time.] |
| **USN\_REASON\_FILE\_DELETE**  0x00000200 | 文件或目录已被删除。  The file or directory was deleted. |
| **USN\_REASON\_EA\_CHANGE**  0x00000400 | 文件或目录的扩展属性已经被更改。  基于windows的应用程序不能访问这些NTFS文件系统属性。  [The file's or directory's extended attributes were changed.  These NTFS file system attributes are not accessible to Windows-based applications.] |
| **USN\_REASON\_SECURITY\_CHANGE**  0x00000800 | 文件或目录的访问权限已经被更改。  [The access rights to the file or directory was changed.] |
| **USN\_REASON\_RENAME\_OLD\_NAME**  0x00001000 | 文件或目录已经被重命名(以前的名称)。即USN\_RECORD\_V2结构中的文件名是以前的名称。  [The file or directory was renamed.(previous name).the file name  in the **USN\_RECORD\_V2** structure is the previous name.] |
| **USN\_REASON\_RENAME\_NEW\_NAME**  0x00002000 | 文件或目录已经被重命名(新的名称)。即USN\_RECORD\_V2结构中的文件名是新的名称。  [The file or directory was renamed.(new name). the file name  in the **USN\_RECORD\_V2** structure is the new name.] |
| **USN\_REASON\_INDEXABLE\_CHANGE**  0x00004000 | 用户更改了FILE\_ATTRIBUTE\_NOT\_CONTENT\_INDEXED属性。  也就是说，用户将文件或目录从内容可以索引的地方更改为内容不能索引的地方，反之亦然。内容索引允许通过构建所选内容的数据库来快速搜索数据。  [A user changes the **FILE\_ATTRIBUTE\_NOT\_CONTENT\_INDEXED** attribute.  That is, the user changes the file or directory from one where content can be indexed to one where content cannot be indexed, or vice versa. Content indexing permits rapid searching of data by building a database of selected content.] |
| **USN\_REASON\_BASIC\_INFO\_CHANGE**  0x00008000 | 用户要么更改了一个或多个文件或目录属性(例如只读、隐藏、系统、存档或稀疏属性)，要么更改了一个或多个时间戳。  [A user has either changed one or more file or directory attributes (for example, the read-only, hidden, system, archive, or sparse attribute), or one or more time stamps.] |
| **USN\_REASON\_HARD\_LINK\_CHANGE**  0x00010000 | 文件或目录的NTFS文件系统硬链接被添加或删除。NTFS文件系统硬链接(类似于POSIX硬链接)是可以看到相同文件或目录的几个目录条目之一。  [An NTFS file system hard link was added to or removed from the file or directory.An NTFS file system hard link, similar to a POSIX hard link, is one of several directory entries that see the same file or directory.] |
| **USN\_REASON\_COMPRESSION\_CHANGE**  0x00020000 | 文件或目录的压缩状态已经从非压缩变为压缩/压缩变为非压缩。[The compression state of the file or directory was changed from or to compressed.] |
| 0x00040000  **USN\_REASON\_ENCRYPTION\_CHANGE** | 文件或目录已加密或解密。  [The file or directory was encrypted or decrypted.] |
| **USN\_REASON\_OBJECT\_ID\_CHANGE**  0x00080000 | 文件或目录的对象标识符已更改。  [The object identifier of the file or directory was changed.] |
| **USN\_REASON\_REPARSE\_POINT\_CHANGE**  0x00100000 | 更改文件或目录中包含的重新解析点，或从文件或目录中添加或删除重新解析点。  [The reparse point contained in the file or directory was changed, or a reparse point was added to or deleted from the file or directory.] |
| **USN\_REASON\_STREAM\_CHANGE**  0x00200000 | 文件的命名流已经被添加或删除，或命名流已重命名。  [A named stream has been added to or removed from the file, or a named stream has been renamed.] |
| **USN\_REASON\_TRANSACTED\_CHANGE**  0x00400000 | 通过TxF事务给定的流已经被修改。  [The given stream is modified through a TxF transaction.] |
| **USN\_REASON\_INTEGRITY\_CHANGE**  0x00800000 | 用户更改了给定流的FILE\_ATTRIBUTE\_INTEGRITY\_STREAM属性的状态。  在ReFS文件系统上，完整性流维护该流的所有数据的校验和，以便在读写操作期间验证文件的内容。  [A user changed the state of the **FILE\_ATTRIBUTE\_INTEGRITY\_STREAM** attribute for the given stream.  On the ReFS file system, integrity streams maintain a checksum of all data for that stream, so that the contents of the file can be validated during read or write operations.] |
| **USN\_REASON\_CLOSE**  0x80000000 | 文件或目录已关闭。  [The file or directory was closed.] |
|  |  |

参考来源：<http://msdn.microsoft.com/en-us/library/aa365722.aspx>

**Source Information**

|  |  |
| --- | --- |
| Value | Meaning |
| 0x00 | 正常事件  Normal Event |
| **USN\_SOURCE\_DATA\_MANAGEMENT**  0x00000001 | 该操作提供有关操作系统更改文件或目录的信息。  一个典型的用法是，远程存储系统将数据从外部存储移动到本地存储。远程存储是分级存储管理软件。这样的操作通常至少会将USN\_REASON\_DATA\_OVERWRITE标志添加到USN记录中。然而，从用户的角度来看，数据并没有改变。通过在SourceInfo成员中注意USN\_SOURCE\_DATA\_MANAGEMENT，您可以确定虽然对项执行了写操作，但是数据没有更改。  [The operation provides information about a change to the file or directory made by the operating system.  A typical use is when the Remote Storage system moves data from external to local storage. Remote Storage is the hierarchical storage management software. Such a move usually at a minimum adds the **USN\_REASON\_DATA\_OVERWRITE** flag to a USN record. However, the data has not changed from the user's point of view. By noting **USN\_SOURCE\_DATA\_MANAGEMENT** in the **SourceInfo** member, you can determine that although a write operation is performed on the item, data has not changed.] |
| **USN\_SOURCE\_AUXILIARY\_DATA**  0x00000002 | 该操作将私有数据流添加到文件或目录中。  一个例子可能是添加校验和信息的病毒检测器。  当病毒检测器修改项目时，系统生成USN记录。  USN\_SOURCE\_AUXILIARY\_DATA表示修改并没有更改应用程序数据。 [The operation adds a private data stream to a file or directory.  An example might be a virus detector adding checksum information.  As the virus detector modifies the item, the system generates USN records.  **USN\_SOURCE\_AUXILIARY\_DATA** indicates that the modifications did not change the application data.] |
| **USN\_SOURCE\_REPLICATION\_MANAGEMENT**  0x00000004 | 该操作创建或更新复制文件的内容。  [The operation creates or updates the contents of a replicated file.] |
| **USN\_SOURCE\_CLIENT\_REPLICATION\_MANAGEMENT**  0x00000008 | 该操作正在修改客户机系统上的一个文件，以匹配云中存在的同一文件的内容。  [The operation is modifying a file on client systems to match the contents of the same file that exists in the cloud.] |
|  |  |

参考来源：<http://msdn.microsoft.com/en-us/library/aa365722.aspx>

**File Attribute**

|  |  |
| --- | --- |
| Value | Meaning |
| **FILE\_ATTRIBUTE\_READONLY**  1 (0x1) | 只读文件。应用程序可以读取文件，但不能写入或删除它。在目录上不启用此属性。有关更多信息，请参见无法在Windows Server 2003、Windows XP、Windows Vista或Windows 7中查看或更改文件夹的只读或系统属性。  [A file that is read-only. Applications can read the file, but cannot write to it or delete it. This attribute is not honored on directories. For more information, see [You cannot view or change the Read-only or the System attributes of folders in Windows Server 2003, in Windows XP, in Windows Vista or in Windows 7](https://go.microsoft.com/fwlink/p/?linkid=125896).] |
| **FILE\_ATTRIBUTE\_HIDDEN**  2 (0x2) | 文件或目录是隐藏的。它不包括在普通目录清单中。  [The file or directory is hidden. It is not included in an ordinary directory listing.] |
| **FILE\_ATTRIBUTE\_SYSTEM**  4 (0x4) | 操作系统使用一部分的文件或目录，或只使用。  [A file or directory that the operating system uses a part of, or uses exclusively.] |
| **FILE\_ATTRIBUTE\_DIRECTORY**  16 (0x10) | 标识目录的句柄。  [The handle that identifies a directory.] |
| **FILE\_ATTRIBUTE\_ARCHIVE**  32 (0x20) | 存档文件或目录。应用程序通常使用此属性标记要备份或删除的文件。  [An archive file or directory. Applications typically use this attribute to mark files for backup or removal . ] |
| **FILE\_ATTRIBUTE\_DEVICE**  64 (0x40) | 此值保留给系统使用。  [This value is reserved for system use.] |
| **FILE\_ATTRIBUTE\_NORMAL**  128 (0x80) | 没有设置其他属性的文件。此属性仅在单独使用时有效。  [A file that does not have other attributes set. This attribute is valid only when used alone.] |
| **FILE\_ATTRIBUTE\_TEMPORARY**  256 (0x100) | 用于临时存储的文件。如果有足够的高速缓存可用，文件系统会避免将数据写回海量存储，因为通常情况下，应用程序会在句柄关闭后删除临时文件。在这种情况下，系统可以完全避免写入数据。否则，数据将在句柄关闭后写入。  [A file that is being used for temporary storage. File systems avoid writing data back to mass storage if sufficient cache memory is available, because typically, an application deletes a temporary file after the handle is closed. In that scenario, the system can entirely avoid writing the data. Otherwise, the data is written after the handle is closed.] |
| **FILE\_ATTRIBUTE\_SPARSE\_FILE**  512 (0x200) | 一个稀疏文件。  [A file that is a sparse file.] |
| **FILE\_ATTRIBUTE\_REPARSE\_POINT**  1024 (0x400) | 具有关联重解析点的文件或目录，或具有符号链接的文件。  [A file or directory that has an associated reparse point, or a file that is a symbolic link.] |
| **FILE\_ATTRIBUTE\_COMPRESSED**  2048 (0x800) | 被压缩的文件或目录。对于一个文件，文件中的所有数据都被压缩。对于目录，对于新创建的文件和子目录，压缩是缺省值。  [A file or directory that is compressed. For a file, all of the data in the file is compressed. For a directory, compression is the default for newly created files and subdirectories.] |
| **FILE\_ATTRIBUTE\_OFFLINE**  4096 (0x1000) | 此属性指示文件数据被物理移动到脱机存储。此属性用于远程存储，即分层存储管理软件。应用程序不应随意更改此属性。  [This attribute indicates that the file data is physically moved to offline storage.This attribute is used by Remote Storage, which is the hierarchical storage management software. Applications should not arbitrarily change this attribute.] |
| **FILE\_ATTRIBUTE\_NOT\_CONTENT\_INDEXED**  8192 (0x2000) | 内容索引服务不为文件或目录建立索引。  [The file or directory is not to be indexed by the content indexing service.] |
| **FILE\_ATTRIBUTE\_ENCRYPTED**  16384 (0x4000) | 加密的文件或目录。对于一个文件，文件中的所有数据流都是加密的。对于目录，对于新创建的文件和子目录，加密是缺省值。 [A file or directory that is encrypted. For a file, all data streams in the file are encrypted. For a directory, encryption is the default for newly created files and subdirectories.] |
| **FILE\_ATTRIBUTE\_INTEGRITY\_STREAM**  32768 (0x8000) | 目录或用户数据流配置为完整性(仅在参考卷上支持)。它不包括在普通目录清单中。如果文件被重命名，则完整性设置将与该文件保持一致。如果复制了一个文件，如果源文件或目标目录具有完整性集，则目标文件将具有完整性集。Windows Server 2008 R2、Windows 7、Windows Server 2008、Windows Vista、Windows Server 2003和Windows XP不支持此标志，直到Windows Server 2012才支持此标志。  [The directory or user data stream is configured with integrity (only supported on ReFS volumes). It is not included in an ordinary directory listing. The integrity setting persists with the file if it's renamed. If a file is copied the destination file will have integrity set if either the source file or destination directory have integrity set. **Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP:** This flag is not supported until Windows Server 2012.] |
| **FILE\_ATTRIBUTE\_VIRTUAL**  65536 (0x10000) | 此值保留给系统使用。  [This value is reserved for system use.] |
| **FILE\_ATTRIBUTE\_NO\_SCRUB\_DATA**  131072 (0x20000) | 用户数据流不能被后台数据完整性扫描器(即scrubber)读取。当在目录上设置时，它只提供继承。此标志仅支持存储空间和引用卷。它不包括在普通目录清单中。Windows Server 2008 R2、Windows 7、Windows Server 2008、Windows Vista、Windows Server 2003和Windows XP不支持此标志，直到Windows 8和Windows Server 2012才支持此标志。  [The user data stream not to be read by the background data integrity scanner (AKA scrubber). When set on a directory it only provides inheritance. This flag is only supported on Storage Spaces and ReFS volumes. It is not included in an ordinary directory listing.**Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP:** This flag is not supported until Windows 8 and Windows Server 2012.] |
| **FILE\_ATTRIBUTE\_RECALL\_ON\_OPEN**  262144 (0x40000) | 此属性只出现在目录枚举类中(FILE\_DIRECTORY\_INFORMATION、FILE\_BOTH\_DIR\_INFORMATION等)。设置此属性时，表示文件或目录在本地系统上没有物理表示;该项目是虚拟的。打开项目会比正常情况下更毫时间/资源等，例如，它会导致至少一部分项目从远程储槽提取。  [This attribute only appears in directory enumeration classes (FILE\_DIRECTORY\_INFORMATION, FILE\_BOTH\_DIR\_INFORMATION, etc.). When this attribute is set, it means that the file or directory has no physical representation on the local system; the item is virtual. Opening the item will be more expensive than normal, e.g. it will cause at least some of it to be fetched from a remote store.] |
| **FILE\_ATTRIBUTE\_RECALL\_ON\_DATA\_ACCESS**  4194304 (0x400000) | 设置此属性时，意味着文件或目录在本地没有完全显示。对于一个文件来说，这意味着它的数据并不都在本地存储上(例如，它可能是稀疏的，一些数据仍然在远程存储中)。对于目录，这意味着从另一个位置虚拟化目录内容。读取文件/枚举目录将比正常情况下更耗时间/资源等，例如，它将导致至少部分文件/目录内容从远程存储中获取。只有内核模式调用者可以设置这个位。  [When this attribute is set, it means that the file or directory is not fully present locally. For a file that means that not all of its data is on local storage (e.g. it may be sparse with some data still in remote storage). For a directory it means that some of the directory contents are being virtualized from another location. Reading the file / enumerating the directory will be more expensive than normal, e.g. it will cause at least some of the file/directory content to be fetched from a remote store. Only kernel-mode callers can set this bit.] |
|  |  |

参考来源：<http://msdn.microsoft.com/en-us/library/gg258117.aspx>

**第三部分 读取变更记录表（$UsnJrnl）的记录**

要读取$UsnJrnl的记录需要按照以下步骤进行：

1、判断卷标的格式；

2、取得卷标的句柄；

3、从卷中取得日志；

4、读取日志的信息；

5、枚举所有的数据；

6、关闭日志；

7、关闭卷标。

**一、判断卷标的格式**

由于只有NTFS文件系统才支持$UsnJrnl日志，因此在读取日志前需要判断卷标的格式。API如下：

|  |
| --- |
| C++ |
| BOOL GetVolumeInformationA(  LPCSTR lpRootPathName,  LPSTR lpVolumeNameBuffer,  DWORD nVolumeNameSize,  LPDWORD lpVolumeSerialNumber,  LPDWORD lpMaximumComponentLength,  LPDWORD lpFileSystemFlags,  LPSTR lpFileSystemNameBuffer,  DWORD nFileSystemNameSize  ); |

参考：<https://docs.microsoft.com/en-us/windows/desktop/api/FileAPI/nf-fileapi-getvolumeinformationa>

|  |  |
| --- | --- |
| lpRootPathName | 卷标名，例如：本地磁盘“C:\”或者网络共享盘“\192.168.1.3\MyShare”，如果lpRootPathName设置成NULL，则使用当前目录所在的盘符（例如：D:\） |
| lpVolumeNameBuffer | 指向缓冲区的指针，该缓冲区接收指定卷的名称。缓冲区大小由nVolumeNameSize参数指定。 |
| nVolumeNameSize | 卷名缓冲区的长度，在TCHARs中。最大缓冲区大小为MAX\_PATH（260）+1。如果没有提供卷名称缓冲区（lpVolumeNameBuffer），则忽略此参数。 |
| lpVolumeSerialNumber | 指向接收卷序列号的变量的指针。如果不需要序列号，则此参数可以为空（NULL）。此函数返回格式化硬盘时操作系统分配的卷序列号。 |
| lpMaximumComponentLength |  |
| lpFileSystemFlags | 指向一个变量的指针，该变量接收与指定文件系统相关联的标志。此参数可以是以下**一个或多个标志**。但是，FILE\_FILE\_COMPRESSION和FILE\_VOL\_IS\_COMPRESSED是互斥的。 |
| lpFileSystemNameBuffer | 指向缓冲区的指针，该缓冲区接收文件系统的名称，例如FAT文件系统或NTFS文件系统。缓冲区大小由nFileSystemNameSize参数指定。 |
| nFileSystemNameSize | 文件系统名称缓冲区的长度，在TCHARs中。最大缓冲区大小为MAX\_PATH+1。  如果不提供文件系统名称缓冲区（lpFileSystemNameBuffer），则忽略此参数。 |
|  |  |

**lpFileSystemFlags**

|  |  |
| --- | --- |
| Value | Meaning |
| **FILE\_CASE\_SENSITIVE\_SEARCH**  0x00000001 | 指定的卷支持区分大小写的文件名。  The specified volume supports case-sensitive file names. |
| **FILE\_CASE\_PRESERVED\_NAMES**  0x00000002 | 指定的卷在磁盘上放置名称时支持保留文件名大小写。  The specified volume supports preserved case of file names when it places a name on disk. |
| **FILE\_UNICODE\_ON\_DISK**  0x00000004 | 指定卷支持Unicode文件名称出现在磁盘上。  The specified volume supports Unicode in file names as they appear on disk. |
| **FILE\_PERSISTENT\_ACLS**  0x00000008 | 指定的卷保存并强制访问控制列表(ACL)。例如，NTFS文件系统保存和强制ACL，而FAT文件系统不保存和强制ACL。  The specified volume preserves and enforces access control lists (ACL). For example, the NTFS file system preserves and enforces ACLs, and the FAT file system does not. |
| **FILE\_FILE\_COMPRESSION**  0x00000010 | 指定的卷支持基于文件的压缩。  The specified volume supports file-based compression. |
| **FILE\_VOLUME\_QUOTAS**  0x00000020 | 指定的卷支持磁盘配额。  The specified volume supports disk quotas. |
| **FILE\_SUPPORTS\_SPARSE\_FILES**  0x00000040 | 指定的卷支持稀疏文件。  The specified volume supports sparse files. |
| **FILE\_SUPPORTS\_REPARSE\_POINTS**  0x00000080 | 指定的卷支持重新解析点。  The specified volume supports reparse points.  **ReFS:** ReFS supports reparse points but does not index  them so[FindFirstVolumeMountPoint](https://msdn.microsoft.com/2ac3c22d-b8dd-43d8-a7af-877b54e42d9d) and [FindNextVolumeMountPoint](https://msdn.microsoft.com/299e2fed-74d8-4008-b593-981c52016532)  will not function as expected. |
| **FILE\_VOLUME\_IS\_COMPRESSED**  0x00008000 | 指定的卷是压缩卷，例如，双空间卷。  The specified volume is a compressed volume, for example, a DoubleSpace volume. |
| **FILE\_SUPPORTS\_OBJECT\_IDS**  0x00010000 | 指定的卷支持对象标识符。  The specified volume supports object identifiers. |
| **FILE\_SUPPORTS\_ENCRYPTION**  0x00020000 | 指定的卷支持加密的文件系统(EFS)。  The specified volume supports the Encrypted File System (EFS). For more information, see[File Encryption](https://msdn.microsoft.com/5f20109f-727d-44a9-90a1-0adc19b00d28). |
| **FILE\_NAMED\_STREAMS**  0x00040000 | 指定的卷支持命名流。  The specified volume supports named streams. |
| **FILE\_READ\_ONLY\_VOLUME**  0x00080000 | 指定的卷是只读的。  The specified volume is read-only. |
| **FILE\_SEQUENTIAL\_WRITE\_ONCE**  0x00100000 | 指定的卷支持单个顺序写入。  The specified volume supports a single sequential write. |
| **FILE\_SUPPORTS\_TRANSACTIONS**  0x00200000 | 指定的卷支持事务。  The specified volume supports transactions. For more information, see [About KTM](https://msdn.microsoft.com/85a79698-a1ae-45a4-805e-25175034fa65). |
| **FILE\_SUPPORTS\_HARD\_LINKS**  0x00400000 | 指定的卷支持硬链接。  The specified volume supports hard links. For more information, see[Hard Links and Junctions](https://msdn.microsoft.com/f9e40a86-a4a6-4524-8045-312da72dc655).  **Windows Server2008, WindowsVista, Windows Server2003 and WindowsXP:**This value is not supported until Windows Server2008R2 and Windows7. |
| **FILE\_SUPPORTS\_EXTENDED\_ATTRIBUTES**  0x00800000 | 指定的卷支持扩展属性。扩展属性是特定于应用程序的元数据，应用程序可以将其与文件关联，而不是文件数据的一部分。  The specified volume supports extended attributes. An extended attribute is a piece of application-specific metadata that an application can associate with a file and is not part of the file's data.  **Windows Server2008, WindowsVista, Windows Server2003 and WindowsXP:**This value is not supported until Windows Server2008R2 and Windows7. |
| **FILE\_SUPPORTS\_OPEN\_BY\_FILE\_ID**  0x01000000 | 文件系统支持通过FileID打开。  The file system supports open by FileID.  For more information, see[FILE\_ID\_BOTH\_DIR\_INFO](https://msdn.microsoft.com/d7011ea4-e70a-4c03-a715-6144ce0c7029).  **Windows Server2008, WindowsVista, Windows Server2003 and WindowsXP:**This value is not supported until Windows Server2008R2 and Windows7. |
| **FILE\_SUPPORTS\_USN\_JOURNAL**  0x02000000 | 指定的卷支持更新序列号(USN)日志。  The specified volume supports update sequence number (USN) journals. For more information, see[Change Journal Records](https://msdn.microsoft.com/c41aa3a8-c8d8-4bf2-9bbb-d6a6a556c5e4).  **Windows Server2008, WindowsVista, Windows Server2003 and WindowsXP:**This value is not supported until Windows Server2008R2 and Windows7. |
| **FILE\_DAX\_VOLUME**  0x20000000 | 指定的卷是直接访问(DAX)卷。  The specified volume is a direct access (DAX) volume.  **Note**  This flag was introduced in Windows 10, version 1607. |
|  |  |

|  |  |
| --- | --- |
| 环境要求 | |
| Minimum supported client | Windows XP [desktop apps | UWP apps] |
| Minimum supported server | Windows Server 2003 [desktop apps | UWP apps] |
| Target Platform | Windows |
| Header | fileapi.h (include Windows.h) |
| Library | Kernel32.lib |
| DLL | Kernel32.dll |
|  |  |

|  |
| --- |
| 备注 |
| 如果函数执行，返回非零值，否则返回零。要获执行失败的错误信息，请调用GetLastError。  当用户试图获取关于没有软盘的软盘驱动器或没有光盘驱动器的CD-ROM驱动器的信息时，系统将显示一个消息框，以便用户分别插入软盘或光盘。要防止系统显示此消息框，请使用SEM\_FAILCRITICALERRORS调用SetErrorMode函数。在Win7及以上的操作系统，支持FILE\_SUPPORTS\_USN\_JOURNAL标志，这意味着可以根据lpFileSystemFlags参数判断是否可以读取$UsnJrnl日志记录。 |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  TCHAR fsName[MAX\_PATH] = { 0 };  DWORD fsFlags = 0;  BOOL bOk = GetVolumeInformation(\_T("G:\\"), NULL, 0, NULL, NULL, &fsFlags, fsName, MAX\_PATH);  if (bOk)  {  printf("文件系统名: %s\n", fsName);  //卷标所属文件系统判断方式1（仅支持Win7及以上操作系统）  //注意返回的文件系统标志可能是多个，因此必须进行&操作。  if ((fsFlags & FILE\_SUPPORTS\_USN\_JOURNAL) == FILE\_SUPPORTS\_USN\_JOURNAL)  {  printf("该驱动盘是NTFS格式\n");  }  //卷标所属文件系统判断方式2（传统判断方式）  if (strcmp(fsName, "NTFS") == 0)  {  printf("该驱动盘是NTFS格式\n");  }  else  {  printf("该驱动盘非NTFS格式\n");  }  } |

**二、取得卷标的句柄**

取得卷标的句柄，需要借助CreateFile API，该函数把一切输入源都当成设备，因此其支持打开的设备种类非常多，例如：光驱、硬盘、邮件、串口、并口、控制台、命名管道、匿名管道等等。API如下:

|  |
| --- |
| C++ |
| HANDLE CreateFileA(  LPCSTR lpFileName,  DWORD dwDesiredAccess,  DWORD dwShareMode,  LPSECURITY\_ATTRIBUTES lpSecurityAttributes,  DWORD dwCreationDisposition,  DWORD dwFlagsAndAttributes,  HANDLE hTemplateFile  ); |

参考：<https://docs.microsoft.com/zh-cn/windows/desktop/api/fileapi/nf-fileapi-createfilea>

|  |  |
| --- | --- |
| lpFileName | 要创建或打开的文件或设备的名称。您可以在这个名称中使用前斜杠(/)或后斜杠(\)。在这个函数的ANSI版本中，名称被限制为MAX\_PATH字符。要将这个限制扩展到32,767个宽字符，请调用函数的Unicode版本并在路径前加上“\?”（Windows10开始已经取消这个限制）。  The name of the file or device to be created or opened. You may use either forward slashes (/) or backslashes () in this name.In the ANSI version of this function, the name is limited to **MAX\_PATH** characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\?" to the path.  For more information, see [Naming Files, Paths, and Namespaces](https://msdn.microsoft.com/121cd5b2-e6fd-4eb4-99b4-b652d27b53e8).For information on special device names, see [Defining an MS-DOS Device Name](https://msdn.microsoft.com/7d802e9f-dc09-4e3d-b064-e9b57af396e2).To create a file stream, specify the name of the file, a colon, and then the name of the stream. For more information, see [File Streams](https://msdn.microsoft.com/41dda6f1-a6d1-4e76-94f3-a72f9e491bee).  **Tip**  Starting with Windows 10, version 1607, for the unicode version of this function (**CreateFileW**), you can opt-in to remove the **MAX\_PATH** limitation without prepending "\\?\". See the "Maximum Path Length Limitation" section of [**Naming Files, Paths, and Namespaces**](https://msdn.microsoft.com/121cd5b2-e6fd-4eb4-99b4-b652d27b53e8) for details. |
| dwDesiredAccess | 对文件或设备的请求访问，可以概括为读、写、两者都是或两者都不是零。最常用的值是GENERIC\_READ、GENERIC\_WRITE或两者都有(GENERIC\_READ | GENERIC\_WRITE)。如果该参数为零，应用程序可以在不访问该文件或设备的情况下查询某些元数据，比如文件、目录或设备属性，即使GENERIC\_READ访问被拒绝。  The requested access to the file or device, which can be summarized as read, write, both or neither zero).The most commonly used values are **GENERIC\_READ**, **GENERIC\_WRITE**, or both (GENERIC\_READ | GENERIC\_WRITE). For more information, see [Generic Access Rights](https://msdn.microsoft.com/e18cede9-9bf7-4866-850b-5d7fa43a5b0f), [File Security and Access Rights](https://msdn.microsoft.com/991d7d94-fae7-406f-b2e3-dee811279366), [File Access Rights Constants](https://msdn.microsoft.com/c534e853-b61f-414d-befe-8d3c4bf08d22), and [ACCESS\_MASK](https://msdn.microsoft.com/f115ee54-3333-4109-8004-d71904a7a943).  If this parameter is zero, the application can query certain metadata such as file, directory, or device attributes without accessing that file or device, even if **GENERIC\_READ** access would have been denied.  You cannot request an access mode that conflicts with the sharing mode that is specified by the *dwShareMode* parameter in an open request that already has an open handle.  For more information, see the Remarks section of this topic and [Creating and Opening Files](https://msdn.microsoft.com/094cac29-c66d-409e-8928-878dc693d393). |
| dwShareMode | 请求的文件或设备的共享模式，可读、可写、可同时读、可删除、可全部共享或不共享(请参阅下表)。对属性或扩展属性的访问请求不受此标志的影响。  The requested sharing mode of the file or device, which can be read, write, both, delete, all of these, or none (refer to the following table). Access requests to attributes or extended attributes are not affected by this flag.  If this parameter is zero and **CreateFile** succeeds, the file or device cannot be shared and cannot be opened again until the handle to the file or device is closed. For more information, see the Remarks section.  You cannot request a sharing mode that conflicts with the access mode that is specified in an existing request that has an open handle. **CreateFile** would fail and the [GetLastError](https://msdn.microsoft.com/d852e148-985c-416f-a5a7-27b6914b45d4) function would return **ERROR\_SHARING\_VIOLATION**.  To enable a process to share a file or device while another process has the file or device open, use a compatible combination of one or more of the following values. For more information about valid combinations of this parameter with the *dwDesiredAccess*parameter, see [Creating and Opening Files](https://msdn.microsoft.com/094cac29-c66d-409e-8928-878dc693d393).  **Note**  The sharing options for each open handle remain in effect until that handle is closed, regardless of process context. |
| lpSecurityAttributes | 一个指向SECURITY\_ATTRIBUTES结构的指针，该结构包含两个独立但相关的数据成员:一个可选的安全描述符和一个布尔值，该值决定返回的句柄是否可以由子进程继承。这个参数可以是NULL。如果该参数为NULL，则CreateFile返回的句柄不能被应用程序可能创建的任何子进程继承，与返回句柄关联的文件或设备将获得默认的安全描述符。  A pointer to a [SECURITY\_ATTRIBUTES](https://msdn.microsoft.com/56b5b350-f4b7-47af-b5f8-6a35f32c1009) structure that contains two separate but related data members: an optional security descriptor, and a Boolean value that determines whether the returned handle can be inherited by child processes.  This parameter can be **NULL**.If this parameter is **NULL**, the handle returned by **CreateFile** cannot be inherited by any child processes the application may create and the file or device associated with the returned handle gets a default security descriptor.  The **lpSecurityDescriptor** member of the structure specifies a [SECURITY\_DESCRIPTOR](https://msdn.microsoft.com/653992aa-4e32-4187-b3ac-727e82bfe0b6) for a file or device. If this member is **NULL**, the file or device associated with the returned handle is assigned a default security descriptor.  **CreateFile** ignores the **lpSecurityDescriptor** member when opening an existing file or device, but continues to use the **bInheritHandle** member.  The **bInheritHandle**member of the structure specifies whether the returned handle can be inherited.  For more information, see the Remarks section. |
| dwCreationDisposition | 对存在或不存在的文件或设备采取的操作。对于文件以外的设备，该参数通常设置为**OPEN\_EXISTING**.。有关更多信息，请参见备注部分。该参数必须是以下值之一，不能组合。  An action to take on a file or device that exists or does not exist.  For devices other than files, this parameter is usually set to **OPEN\_EXISTING**.  For more information, see the Remarks section.  This parameter must be one of the following values, which cannot be combined: |
| dwFlagsAndAttributes | 文件或设备属性和标志，FILE\_ATTRIBUTE\_NORMAL是文件最常见的默认值。这个参数可以包含可用文件属性的任何组合(FILE\_ATTRIBUTE\_\*)。该参数还可以包含用于控制文件或设备缓存行为、访问模式和其他特殊用途标志的标志组合(FILE\_FLAG\_\*)。  The file or device attributes and flags, **FILE\_ATTRIBUTE\_NORMAL** being the most common default value for files.  This parameter can include any combination of the available file attributes (**FILE\_ATTRIBUTE\_\***). All other file attributes override**FILE\_ATTRIBUTE\_NORMAL**.  This parameter can also contain combinations of flags (**FILE\_FLAG\_**) for control of file or device caching behavior, access modes, and other special-purpose flags. These combine with any **FILE\_ATTRIBUTE\_** values.  This parameter can also contain Security Quality of Service (SQOS) information by specifying the **SECURITY\_SQOS\_PRESENT** flag. Additional SQOS-related flags information is presented in the table following the attributes and flags tables.  **Note**  When **CreateFile** opens an existing file, it generally combines the file flags with the file attributes of the existing file, and ignores any file attributes supplied as part of *dwFlagsAndAttributes*. Special cases are detailed in [**Creating and Opening Files**](https://msdn.microsoft.com/094cac29-c66d-409e-8928-878dc693d393).    Some of the following file attributes and flags may only apply to files and not necessarily all other types of devices that **CreateFile**can open. For additional information, see the Remarks section of this topic and [Creating and Opening Files](https://msdn.microsoft.com/094cac29-c66d-409e-8928-878dc693d393).  For more advanced access to file attributes, see [SetFileAttributes](https://msdn.microsoft.com/3d5400c3-555f-44fc-9470-52a36d04d90b). For a complete list of all file attributes with their values and descriptions, see [File Attribute Constants](https://msdn.microsoft.com/ed9a73d2-7fb6-4fb7-97f6-4dbf89e2f156). |
| hTemplateFile | 具有GENERIC\_READ访问权限的模板文件的有效句柄。模板文件为正在创建的文件提供文件属性和扩展属性。这个参数可以是NULL。当打开现有文件时，CreateFile将忽略此参数。  A valid handle to a template file with the **GENERIC\_READ** access right. The template file supplies file attributes and extended attributes for the file that is being created.  This parameter can be **NULL**.When opening an existing file, **CreateFile** ignores this parameter.When opening a new encrypted file, the file inherits the discretionary access control list from its parent directory. For additional information, see [File Encryption](https://msdn.microsoft.com/5f20109f-727d-44a9-90a1-0adc19b00d28). |
|  |  |

dwFlagsAndAttributes（注意：dwFlagsAndAttributes参数前面已经介绍，不再重复）

|  |  |
| --- | --- |
| Value | Meaning |
| **FILE\_FLAG\_BACKUP\_SEMANTICS**  0x02000000 | 正在为备份或还原操作打开或创建文件。当进程具有SE\_BACKUP\_NAME和SE\_RESTORE\_NAME特权时，系统确保调用进程覆盖文件安全检查。有关更多信息，请查看令牌中的特权（[Changing Privileges in a Token](https://msdn.microsoft.com/b8e47d04-07c1-4d57-8209-6b0c397476e5)）。  必须设置此标志才能获得目录的句柄。目录句柄可以传递给某些函数，而不是文件句柄。  The file is being opened or created for a backup or restore operation. The system ensures that the calling process overrides file security checks when the process has **SE\_BACKUP\_NAME** and **SE\_RESTORE\_NAME** privileges. For more information, see[Changing Privileges in a Token](https://msdn.microsoft.com/b8e47d04-07c1-4d57-8209-6b0c397476e5).  You must set this flag to obtain a handle to a directory. A directory handle can be passed to some functions instead of a file handle. For more information, see the Remarks section. |
| **FILE\_FLAG\_DELETE\_ON\_CLOSE**  0x04000000 | 文件的所有句柄都关闭后立即删除，其中包括指定的句柄和任何其他打开或复制的句柄。如果文件存在打开句柄，则调用将失败，除非所有句柄都使用FILE\_SHARE\_DELETE共享模式打开。除非指定了FILE\_SHARE\_DELETE共享模式，否则对文件的后续打开请求将失败。  The file is to be deleted immediately after all of its handles are closed, which includes the specified handle and any other open or duplicated handles.  If there are existing open handles to a file, the call fails unless they were all opened with the **FILE\_SHARE\_DELETE** share mode.Subsequent open requests for the file fail, unless the **FILE\_SHARE\_DELETE** share mode is specified. |
| **FILE\_FLAG\_NO\_BUFFERING**  0x20000000 | 正在打开的文件或设备没有用于数据读写的系统缓存。此标志不影响硬盘缓存或内存映射文件。对于成功地处理使用FILE\_FLAG\_NO\_BUFFERING标志在CreateFile中打开的文件有严格的要求（必须按扇区大小读写），有关详细信息，请参见文件缓冲（[File Buffering](https://msdn.microsoft.com/ae1e5d0f-9b55-4aae-8402-b9c8e33d9363)）。  The file or device is being opened with no system caching for data reads and writes. This flag does not affect hard disk caching or memory mapped files.  There are strict requirements for successfully working with files opened with**CreateFile** using the **FILE\_FLAG\_NO\_BUFFERING** flag, for details see[File Buffering](https://msdn.microsoft.com/ae1e5d0f-9b55-4aae-8402-b9c8e33d9363). |
| **FILE\_FLAG\_OPEN\_NO\_RECALL**  0x00100000 | 请求文件数据，但它应该继续位于远程存储中。它不应该被送回本地存储。此标志供远程存储系统使用。  The file data is requested, but it should continue to be located in remote storage. It should not be transported back to local storage. This flag is for use by remote storage systems. |
| **FILE\_FLAG\_OPEN\_REPARSE\_POINT**  0x00200000 | 不会发生正常的重新解析点处理;CreateFile将尝试打开reparse点。当打开一个文件时，返回一个文件句柄，不管控制reparse点的过滤器是否工作。  此标志不能与CREATE\_ALWAYS标志一起使用。如果文件不是reparse点，则此标志被忽略。  Normal [reparse point](https://msdn.microsoft.com/3abb3a08-9a00-43eb-9792-82eab1a25f06) processing will not occur; **CreateFile** will attempt to open the reparse point. When a file is opened, a file handle is returned, whether or not the filter that controls the reparse point is operational.  This flag cannot be used with the **CREATE\_ALWAYS** flag.  If the file is not a reparse point, then this flag is ignored. |
| **FILE\_FLAG\_OVERLAPPED**  0x40000000 | 正在为异步I/O打开或创建文件或设备。当后续I/O操作在此句柄上完成时，在重叠结构中指定的事件将被设置为有信号状态。如果指定了此标志，则文件可用于同时进行读写操作。  The file or device is being opened or created for asynchronous I/O.  When subsequent I/O operations are completed on this handle, the event specified in the [OVERLAPPED](https://msdn.microsoft.com/5037f6b9-e316-483b-a8e2-b58d2587ebd9) structure will be set to the signaled state.  If this flag is specified, the file can be used for simultaneous read and write operations.  If this flag is not specified, then I/O operations are serialized, even if the calls to the read and write functions specify an [OVERLAPPED](https://msdn.microsoft.com/5037f6b9-e316-483b-a8e2-b58d2587ebd9) structure.  For information about considerations when using a file handle created with this flag, see the [Synchronous and Asynchronous I/O Handles](https://docs.microsoft.com/) section of this topic. |
| **FILE\_FLAG\_POSIX\_SEMANTICS**  0x0100000 | 访问将根据POSIX规则进行。这包括允许具有名称的多个文件(仅在特定情况下不同)用于支持该名称的文件系统。使用此选项时要小心，因为使用此标志创建的文件可能无法被为MS-DOS或16位Windows编写的应用程序访问。  Access will occur according to POSIX rules. This includes allowing multiple files with names, differing only in case, for file systems that support that naming. Use care when using this option, because files created with this flag may not be accessible by applications that are written for MS-DOS or 16-bit Windows. |
| **FILE\_FLAG\_RANDOM\_ACCESS**  0x10000000 | 访问是随机的。系统可以将此作为优化文件缓存的提示。  如果文件系统不支持缓存的I/O和**FILE\_FLAG\_NO\_BUFFERING**，则此标志无效。  Access is intended to be random. The system can use this as a hint to optimize file caching.  This flag has no effect if the file system does not support cached I/O and**FILE\_FLAG\_NO\_BUFFERING**.  For more information, see the [Caching Behavior](https://docs.microsoft.com/) section of this topic. |
| **FILE\_FLAG\_SESSION\_AWARE**  0x00800000 | The file or device is being opened with session awareness. If this flag is not specified, then per-session devices (such as a device using RemoteFX USB Redirection) cannot be opened by processes running in session 0. This flag has no effect for callers not in session 0. This flag is supported only on server editions of Windows.  **Windows Server 2008 R2 and Windows Server 2008:**This flag is not supported before Windows Server 2012. |
| **FILE\_FLAG\_SEQUENTIAL\_SCAN**  0x08000000 | 访问从开始到结束都是顺序的。系统可以将此作为优化文件缓存的提示。  如果使用后读(即反向扫描)，则不应使用此标志。  如果文件系统不支持缓存的I/O和**FILE\_FLAG\_NO\_BUFFERING**，则此标志无效。有关更多信息，请参见本主题的缓存行为部分（[Caching Behavior](https://docs.microsoft.com/)）。  Access is intended to be sequential from beginning to end. The system can use this as a hint to optimize file caching.  This flag should not be used if read-behind (that is, reverse scans) will be used.  This flag has no effect if the file system does not support cached I/O and**FILE\_FLAG\_NO\_BUFFERING**.  For more information, see the [Caching Behavior](https://docs.microsoft.com/) section of this topic. |
| **FILE\_FLAG\_WRITE\_THROUGH**  0x80000000 | 写操作不会经过任何中间缓存，它们将直接进入磁盘。有关其他信息，请参见本主题的缓存行为部分（[Caching Behavior](https://docs.microsoft.com/)）。  Write operations will not go through any intermediate cache, they will go directly to disk.For additional information, see the [Caching Behavior](https://docs.microsoft.com/) section of this topic. |
|  |  |

|  |  |
| --- | --- |
| 环境要求 | |
| Minimum supported client | Windows XP [desktop apps | UWP apps] |
| Minimum supported server | Windows Server 2003 [desktop apps | UWP apps] |
| Target Platform | Windows |
| Header | fileapi.h (include Windows.h) |
| Library | Kernel32.lib |
| DLL | Kernel32.dll |
|  |  |

|  |
| --- |
| 备注 |
| 如果函数成功，返回值是指定文件、设备、命名管道或邮件槽的打开句柄。  如果函数失败，返回值为**INVALID\_HANDLE\_VALUE**。要获取扩展的错误信息，请调用GetLastError。  CreateFile最初是专门为文件交互开发的，但后来得到了扩展和增强，包括大多数其他类型的I/O设备和Windows开发人员可用的机制。当应用程序使用CreateFile返回的对象句柄完成时，使用CloseHandle函数关闭句柄释放系统资源。  您可以使用CreateFile函数打开物理磁盘驱动器或卷，它返回一个可以与DeviceIoControl函数一起使用的直接访问存储设备(DASD)句柄。这使您能够直接访问磁盘或卷，例如分区表之类的磁盘元数据。然而，这种类型的访问还将磁盘驱动器或卷暴露给潜在的数据丢失，因为使用这种机制对磁盘的不正确写入可能使操作系统无法访问其内容。为了确保数据的完整性，一定要熟悉DeviceIoControl以及其他API使用直接访问句柄(与文件系统句柄不同)时的行为。  要使直接访问存储设备获得成功，必须满足下列条件：  ✓调用方必须具有管理（administrative privileges）。有关更多信息，请参见使用特权运行（[Running with Special Privileges](https://msdn.microsoft.com/b25db548-d5ab-4276-9b50-36d030909384).）。  ✓dwCreationDisposition参数必须具有OPEN\_EXISTING标志。  ✓当打开卷或软盘时，dwShareMode参数必须具有FILE\_SHARE\_WRITE标志。  注意，dwDesiredAccess参数可以为零，允许应用程序在不访问设备的情况下查询设备属性。这对于应用程序确定软盘驱动器的大小及其支持的格式非常有用，例如，不需要在驱动器中使用软盘。它还可以用于读取统计数据，而不需要更高级别的数据读/写权限。  当打开一个物理驱动器X:时，lpFileName字符串应该是以下形式:“\\.\PhysicalDriveX”。硬盘号码从零开始。例如：“\\.\PhysicalDrive0”打开第一个物理驱动器，“\\.\PhysicalDrive2”打开第三个物理驱动器。  要获取卷的物理驱动器标识符，请打开卷的句柄，并使用[IOCTL\_VOLUME\_GET\_VOLUME\_DISK\_EXTENTS](https://msdn.microsoft.com/8faff037-d815-48f8-8b59-d63f4ff4a746).调用[DeviceIoControl](https://msdn.microsoft.com/1d35c087-6672-4fc6-baa1-a886dd9d3878)函数。此控制代码返回每个卷的一个或多个区段的磁盘号和偏移量;一个卷可以跨多个物理磁盘。打开物理驱动器的示例，请参见[Calling DeviceIoControl](https://msdn.microsoft.com/b4dbda89-effb-43f7-b3cc-774db57862a9).。  当打开卷或可移动媒体驱动器(例如软盘驱动器或闪存拇指驱动器)时，lpFileName字符串应该是以下形式："\\.\X:"。不要使用后面的反斜杠(\)，否则会失败，因为它表示驱动器的根目录。例如：  "\\.\A:" Opens floppy disk drive A（注意打开软盘时卷标后面无反斜杠）  "\\.\H:" Opens the C: volume（注意打开可移动磁盘时卷标后面无反斜杠）  "\\.\C:" Opens the file system of the C: volume. （注意打开硬盘时卷标后面不带反斜杠）  注意:要读写卷的最后几个扇区，必须调用DeviceIoControl并指定FSCTL\_ALLOW\_EXTENDED\_DASD\_IO。这表示文件系统驱动程序在分区读或写调用上不执行任何I/O边界检查。相反，边界检查由设备驱动程序执行。 |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  HANDLE hVol = CreateFile(\_T("\\\\.\\G:"),  GENERIC\_READ | GENERIC\_WRITE,  FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,  NULL,  OPEN\_EXISTING,  FILE\_ATTRIBUTE\_READONLY,  NULL);  if (hVol != INVALID\_HANDLE\_VALUE) //注意用INVALID\_HANDLE\_VALUE判断是否打开成功，不要用NULL判断  {  printf("获取卷的句柄成功，handle:%d", hVol);  }  else  {  printf("获取卷的句柄失败，handle:%d error:%d\n", hVol, GetLastError());  } |

**三、从卷中取得日志**

当取得卷的句柄以后，就可以从其中取得日志了。API如下：

|  |
| --- |
| C++ |
| BOOL DeviceIoControl(  HANDLE hDevice,  DWORD dwIoControlCode,  LPVOID lpInBuffer,  DWORD nInBufferSize,  LPVOID lpOutBuffer,  DWORD nOutBufferSize,  LPDWORD lpBytesReturned,  LPOVERLAPPED lpOverlapped  ); |

参考：<https://docs.microsoft.com/en-us/windows/desktop/api/ioapiset/nf-ioapiset-deviceiocontrol>

|  |  |
| --- | --- |
| hDevice | 要在其上执行操作的设备的句柄。设备通常是一个卷、目录、文件或流。要检索设备句柄，请使用CreateFile函数。  A handle to the device on which the operation is to be performed. The device is typically a volume, directory, file, or stream. To retrieve a device handle, use the [CreateFile](https://msdn.microsoft.com/en-us/library/Aa363858(v=VS.85).aspx) function. For more information, see Remarks. |
| dwIoControlCode | 操作的控制代码。此值标识要执行的特定操作以及要在其上执行该操作的设备类型。有关控制代码的列表，请参见备注。每个控制代码的文档提供了lpInBuffer、nInBufferSize、lpOutBuffer和nOutBufferSize参数的使用细节。  The control code for the operation. This value identifies the specific operation to be performed and the type of device on which to perform it.For a list of the control codes, see Remarks. The documentation for each control code provides usage details for the *lpInBuffer*, *nInBufferSize*, *lpOutBuffer*, and *nOutBufferSize* parameters. |
| lpInBuffer | 指向包含执行操作所需数据的输入缓冲区的指针。该数据的格式取决于dwIoControlCode参数的值。如果dwIoControlCode指定不需要输入数据的操作，则此参数可以为空。  A pointer to the input buffer that contains the data required to perform the operation. The format of this data depends on the value of the *dwIoControlCode* parameter.This parameter  can be **NULL** if *dwIoControlCode* specifies an operation that does not require input data. |
| nInBufferSize | 输入缓冲区的大小，以字节为单位。  The size of the input buffer, in bytes. |
| lpOutBuffer | 指向输出缓冲区的指针，用于接收操作返回的数据。该数据的格式取决于dwIoControlCode参数的值。如果dwIoControlCode指定不返回数据的操作，则此参数可以为空。  A pointer to the output buffer that is to receive the data returned by the operation. The format of this data depends on the value of the *dwIoControlCode* parameter.This parameter  can be **NULL** if *dwIoControlCode* specifies an operation that does not return data. |
| nOutBufferSize | 输出缓冲区的大小，以字节为单位。  The size of the output buffer, in bytes. |
| lpBytesReturned | 指向一个变量的指针，该变量接收以字节为单位存储在输出缓冲区中的数据的大小。  如果输出缓冲区太小，无法接收任何数据，则调用失败，GetLastError返回ERROR\_INSUFFICIENT\_BUFFER, lpBytesReturned为零。  如果输出缓冲区太小，无法容纳所有数据，但是可以容纳一些条目，那么一些驱动程序将返回尽可能多的数据。在这种情况下，调用失败，GetLastError返回ERROR\_MORE\_DATA, lpBytesReturned指示接收的数据量。您的应用程序应该使用相同的操作再次调用DeviceIoControl，指定一个新的起点。  如果llapping为空，则lpBytesReturned不能为空。即使一个操作没有返回任何输出数据，并且lpOutBuffer为NULL,DeviceIoControl也会使用lpBytesReturned。在这样的操作之后，返回的lpBytesReturned的值是没有意义的。  如果lpOverlapped不为空，则lpBytesReturne可以为空。如果该参数不为空，且操作返回数据，则在重叠操作完成之前，返回的lpBytesReturned是无意义的。要检索返回的字节数，可以调用[GetOverlappedResult](https://msdn.microsoft.com/7f999959-9b22-4491-ae2b-a2674d821110)。如果hDevice与I/O完成端口相关联，则可以通过调用[GetQueuedCompletionStatus](https://msdn.microsoft.com/en-us/library/Aa364986(v=VS.85).aspx).检索返回的字节数。  A pointer to a variable that receives the size of the data stored in the output buffer, in bytes.  If the output buffer is too small to receive any data, the call fails, [GetLastError](https://msdn.microsoft.com/d852e148-985c-416f-a5a7-27b6914b45d4) returns **ERROR\_INSUFFICIENT\_BUFFER**, and *lpBytesReturned* is zero.  If the output buffer is too small to hold all of the data but can hold some entries, some drivers will return as much data as fits. In this case, the call fails, [GetLastError](https://msdn.microsoft.com/d852e148-985c-416f-a5a7-27b6914b45d4) returns **ERROR\_MORE\_DATA**, and *lpBytesReturned* indicates the amount of data received. Your application should call **DeviceIoControl** again with the same operation, specifying a new starting point.  If *lpOverlapped* is **NULL**, *lpBytesReturned* cannot be **NULL**. Even when an operation returns no output data and *lpOutBuffer* is **NULL**,**DeviceIoControl** makes use of *lpBytesReturned*. After such an operation, the value of *lpBytesReturned* is meaningless.  If *lpOverlapped* is not **NULL**, *lpBytesReturned* can be **NULL**. If this parameter is not **NULL** and the operation returns data, *lpBytesReturned* is meaningless until the overlapped operation has completed. To retrieve the number of bytes returned, call[GetOverlappedResult](https://msdn.microsoft.com/7f999959-9b22-4491-ae2b-a2674d821110). If *hDevice* is associated with an I/O completion port, you can retrieve the number of bytes returned by calling[GetQueuedCompletionStatus](https://msdn.microsoft.com/en-us/library/Aa364986(v=VS.85).aspx). |
| lpOverlapped | 指向重叠结构的指针。  如果在没有指定FILE\_FLAG\_OVERLAPPED的情况下打开hDevice，则忽略lpOverlapped 。如果hDevice是用FILE\_FLAG\_OVERLAPPED标志打开的，那么该操作将作为一个重叠(异步)操作执行。在这种情况下，lpOverlapped必须指向一个有效的重叠结构，该结构包含一个事件对象的句柄。否则，函数将以不可预知的方式失败。对于重叠操作，DeviceIoControl将立即返回，当操作完成时，事件对象将发出信号。否则，在操作完成或发生错误之前，函数不会返回。  A pointer to an [OVERLAPPED](https://msdn.microsoft.com/5037f6b9-e316-483b-a8e2-b58d2587ebd9) structure.  If *hDevice* was opened without specifying **FILE\_FLAG\_OVERLAPPED**, *lpOverlapped* is ignored.  If *hDevice* was opened with the **FILE\_FLAG\_OVERLAPPED** flag, the operation is performed as an overlapped (asynchronous) operation. In this case, *lpOverlapped* must point to a valid [OVERLAPPED](https://msdn.microsoft.com/5037f6b9-e316-483b-a8e2-b58d2587ebd9) structure that contains a handle to an event object. Otherwise, the function fails in unpredictable ways.  For overlapped operations, **DeviceIoControl** returns immediately, and the event object is signaled when the operation has been completed. Otherwise, the function does not return until the operation has been completed or an error occurs. |
|  |  |

|  |  |
| --- | --- |
| 环境要求 | |
| Minimum supported client | Windows XP [desktop apps | UWP apps] |
| Minimum supported server | Windows Server 2003 [desktop apps | UWP apps] |
| Target Platform | Windows |
| Header | winioctl.h (include Windows.h) |
| Library | Kernel32.lib |
| DLL | Kernel32.dll |
|  |  |

|  |
| --- |
| 备注 |
| **如果操作成功完成，返回值为非零。**  **如果操作失败或挂起，返回值为零。要获取扩展的错误信息，请调用GetLastError。**  要检索设备的句柄，必须使用设备的名称或与设备关联的驱动程序的名称调用[CreateFile](https://msdn.microsoft.com/en-us/library/Aa363858(v=VS.85).aspx)函数。要指定设备名称，请使用以下格式：\\.\设备名称:，例如：[\\.\C](file:///\\.\C):。  DeviceIoControl可以接受特定设备的句柄。例如，要用CreateFile打开逻辑驱动器A:的句柄，请指定\\.\A:。或者，您可以使用名称[\\.\PhysicalDrive0](file:///\\.\PhysicalDrive0)、[\\.\PhysicalDrive1](file:///\\.\PhysicalDrive0)，以此类推，打开系统上物理驱动器的句柄。  在调用CreateFile打开设备驱动程序句柄时，应该指定FILE\_SHARE\_READ和FILE\_SHARE\_WRITE访问标志。但是，当您打开通信资源(例如串行端口)时，必须指定独占访问。当打开设备句柄时，使用其他CreateFile参数如下：  ✓fdwCreate参数必须指定**OPEN\_EXISTING**。  ✓hTemplateFile参数必须为**NULL**。  ✓fdwAttrsAndFlags参数可以指定FILE\_FLAG\_OVERLAPPED，以指示返回的句柄可以用于重叠(异步)I/O操作。  有关支持的控制代码列表，请参阅以下主题：  ✓[Communications Control Codes](https://msdn.microsoft.com/a9aed6bb-05aa-4378-837a-ea7ccda39ba4)  ✓[Device Management Control Codes](https://msdn.microsoft.com/b3a3ffa1-e710-4d96-aff8-5b6876ab032b)  ✓[Directory Management Control Codes](https://msdn.microsoft.com/e2e671c7-ef65-4401-8016-649e86f84fec)  ✓[Disk Management Control Codes](https://msdn.microsoft.com/488a7d32-cbb5-4f32-9655-0aca8ac69640)  ✓[File Management Control Codes](https://msdn.microsoft.com/e27ded4b-d104-4244-b38e-5fed10d32e1e)  ✓[Power Management Control Codes](https://msdn.microsoft.com/027fffdb-62a1-47d8-b69f-c2fcf7f9ac97)  ✓[Volume Management Control Codes](https://msdn.microsoft.com/87f39e1c-3ebf-4c6f-a842-699ec3c45e76) |

下面列出卷标管理的控制代码（Volume Management Control Codes）

|  |  |
| --- | --- |
| Value | Meaning |
| [**FSCTL\_CREATE\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364558(v=VS.85).aspx) | 在目标卷上创建更新序列号(USN)更改日志流，或修改现有更改日志流。  Creates an update sequence number (USN) change journal stream on a target volume, or modifies an existing change journal stream. |
| [**FSCTL\_CSV\_QUERY\_DOWN\_LEVEL\_FILE\_SYSTEM\_CHARACTERISTICS**](https://msdn.microsoft.com/en-us/library/Dn280517(v=VS.85).aspx) | 检索有关CSVFS是其代理的文件系统的信息。  Retrieves information about a file system for which CSVFS is a proxy. |
| [**FSCTL\_DELETE\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364561(v=VS.85).aspx) | 删除卷上的更新序列号(USN)更改日志，或等待更改日志删除通知。  Deletes the update sequence number (USN) change journal on a volume, or waits for notification of change journal deletion. |
| [**FSCTL\_DISMOUNT\_VOLUME**](https://msdn.microsoft.com/en-us/library/Aa364562(v=VS.85).aspx) | 卸载一个卷，不管该卷当前是否正在使用。  Dismounts a volume regardless of whether or not the volume is currently in use. For more information, see the Remarks section. |
| [**FSCTL\_ENUM\_USN\_DATA**](https://msdn.microsoft.com/en-us/library/Aa364563(v=VS.85).aspx) | 枚举两个指定边界之间的更新序列号(USN)数据，以获取主文件表(MFT)记录。（**注**：从MSDN解释看出，枚举USN数据时实际读取的是MFT主表中的记录，即$UsnJrnl中实际上是按照顺序（USN）记录系统对文件的增删改操作，也就是说它从时间的维度（也就是USN）统计变更记录，而生成相应的日志供用户快速检阅）  Enumerates the update sequence number (USN) data between two specified boundaries to obtain master file table (MFT) records. |
| [**FSCTL\_EXTEND\_VOLUME**](https://msdn.microsoft.com/en-us/library/Aa364564(v=VS.85).aspx) | 增加挂载卷的大小。  Increases the size of a mounted volume. |
| [**FSCTL\_GET\_BOOT\_AREA\_INFO**](https://msdn.microsoft.com/en-us/library/Dd405525(v=VS.85).aspx) | 检索卷的启动扇区位置。  Retrieves the locations of boot sectors for a volume.  To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**FSCTL\_GET\_INTEGRITY\_INFORMATION**](https://msdn.microsoft.com/en-us/library/Hh965606(v=VS.85).aspx) | 检索参考卷上文件或目录的完整性状态。  Retrieves the integrity status of a file or directory on a ReFS volume. |
| [**FSCTL\_GET\_NTFS\_VOLUME\_DATA**](https://msdn.microsoft.com/en-us/library/Aa364569(v=VS.85).aspx) | 检索有关指定NTFS文件系统卷的信息。  Retrieves information about the specified NTFS file system volume. |
| [**FSCTL\_GET\_RETRIEVAL\_POINTER\_BASE**](https://msdn.microsoft.com/en-us/library/Dd405526(v=VS.85).aspx) | 返回文件系统相对于卷开始的第一个逻辑簇号(LCN)的扇区偏移量。  Returns the sector offset to the first logical cluster number (LCN) of the file system relative to the start of the volume. To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**FSCTL\_GET\_RETRIEVAL\_POINTERS**](https://msdn.microsoft.com/en-us/library/Aa364572(v=VS.85).aspx) | 给定文件句柄，检索描述特定文件在磁盘上的分配和位置的数据结构，或者，给定卷句柄，检索卷上坏集群的位置。  Given a file handle, retrieves a data structure that describes the allocation and location on disk of a specific file, or, given a volume handle, the locations of bad clusters on a volume. |
| [**FSCTL\_GET\_VOLUME\_BITMAP**](https://msdn.microsoft.com/en-us/library/Aa364573(v=VS.85).aspx) | 检索卷上已占用和可用簇的位图。  Retrieves a bitmap of occupied and available clusters on a volume. |
| [**FSCTL\_IS\_CSV\_FILE**](https://msdn.microsoft.com/en-us/library/Dn280518(v=VS.85).aspx) | 确定文件是存储在CSVFS卷中，还是检索名称空间信息。  Determines whether a file is stored on a CSVFS volume, or retrieves namespace information. |
| [**FSCTL\_IS\_FILE\_ON\_CSV\_VOLUME**](https://msdn.microsoft.com/en-us/library/Dn280519(v=VS.85).aspx) | 确定文件是存储在CSVFS卷中，还是检索名称空间信息。  Determines whether a file is stored on a CSVFS volume, or retrieves namespace information. |
| [**FSCTL\_IS\_VOLUME\_MOUNTED**](https://msdn.microsoft.com/en-us/library/Aa364574(v=VS.85).aspx) | 确定是否挂载指定卷，或指定文件或目录是否位于挂载卷上。  Determines whether the specified volume is mounted, or if the specified file or directory is on a mounted volume. |
| [**FSCTL\_IS\_VOLUME\_OWNED\_BYCSVFS**](https://msdn.microsoft.com/en-us/library/Dn280520(v=VS.85).aspx) | 确定卷是否被CSVFS锁定。  Determines whether a volume is locked by CSVFS. |
| [**FSCTL\_LOCK\_VOLUME**](https://msdn.microsoft.com/en-us/library/Aa364575(v=VS.85).aspx) | 锁定不使用的卷。  Locks a volume if it is not in use. |
| [**FSCTL\_LOOKUP\_STREAM\_FROM\_CLUSTER**](https://msdn.microsoft.com/en-us/library/Ff951637(v=VS.85).aspx) | 给定NTFS卷的句柄或NTFS卷上的文件，返回描述占用指定集群的流的数据结构链。  Given a handle to a NTFS volume or a file on a NTFS volume, returns a chain of data structures that describes streams that occupy the specified clusters. |
| [**FSCTL\_MARK\_HANDLE**](https://msdn.microsoft.com/en-us/library/Aa364576(v=VS.85).aspx) | 将指定的文件或目录及其更改日志记录标记为有关该文件或目录更改的信息。  Marks a specified file or directory and its change journal record with information about changes to that file or directory. |
| [**FSCTL\_MOVE\_FILE**](https://msdn.microsoft.com/en-us/library/Aa364577(v=VS.85).aspx) | 在同一卷内将文件的一个或多个虚拟集群从一个逻辑集群重定位到另一个。此操作在碎片整理期间使用。  Relocates one or more virtual clusters of a file from one logical cluster to another within the same volume. This operation is used during [defragmentation](https://docs.microsoft.com/zh-cn/windows/desktop/FileIO/defragmenting-files). |
| [**FSCTL\_QUERY\_FILE\_SYSTEM\_RECOGNITION**](https://msdn.microsoft.com/en-us/library/Dd442655(v=VS.85).aspx) | 查询卷上的文件系统识别信息。  Queries for file system recognition information on a volume. To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**FSCTL\_QUERY\_REGION\_INFO**](https://msdn.microsoft.com/en-us/library/Dn393706(v=VS.85).aspx) | 检索为支持数据分层的卷定义的存储层区域。  Retrieves the storage tier regions defined for a volume that supports data tiering. To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**FSCTL\_QUERY\_STORAGE\_CLASSES**](https://msdn.microsoft.com/en-us/library/Dn393709(v=VS.85).aspx) | 检索为支持数据分层的卷定义的存储层。  Retrieves the storage tiers defined for a volume that supports data tiering. To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**FSCTL\_QUERY\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364583(v=VS.85).aspx) | 查询关于当前更新序列号(USN)更改日志、其记录和容量的信息。  Queries for information on the current update sequence number (USN) change journal, its records, and its capacity. |
| [**FSCTL\_READ\_FILE\_USN\_DATA**](https://msdn.microsoft.com/en-us/library/Aa364584(v=VS.85).aspx) | 检索指定文件或目录的更新序列号(USN)更改日志信息。  Retrieves the update sequence number (USN) change-journal information for the specified file or directory. |
| [**FSCTL\_READ\_FROM\_PLEX**](https://msdn.microsoft.com/en-us/library/Aa364585(v=VS.85).aspx) | 从指定的plex中读取。  Reads from the specified plex. |
| [**FSCTL\_READ\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364586(v=VS.85).aspx) | 检索两个指定USN值之间的更新序列号(USN)更改日志记录集。  Retrieves the set of update sequence number (USN) change journal records between two specified USN values. |
| [**FSCTL\_REPAIR\_COPIES**](https://msdn.microsoft.com/en-us/library/Hh965608(v=VS.85).aspx) | 通过选择要使用的正确副本来修复数据损坏。  Repair data corruption by selecting the proper copy to use. |
| [**FSCTL\_SET\_INTEGRITY\_INFORMATION**](https://msdn.microsoft.com/en-us/library/Hh965609(v=VS.85).aspx) | 检索参考卷上文件或目录的完整性状态。  Retrieves the integrity status of a file or directory on a ReFS volume. |
| [**FSCTL\_SHRINK\_VOLUME**](https://msdn.microsoft.com/en-us/library/Aa964912(v=VS.85).aspx) | 表示准备好执行收缩操作、提交收缩操作或终止收缩操作的卷。  Signals that the volume is to be prepared to perform the shrink operation, the shrink operation is to be committed, or the shrink operation is to be terminated. |
| [**FSCTL\_UNLOCK\_VOLUME**](https://msdn.microsoft.com/en-us/library/Aa364814(v=VS.85).aspx) | 解锁一个卷。  Unlocks a volume. |
| [**FSCTL\_USN\_TRACK\_MODIFIED\_RANGES**](https://msdn.microsoft.com/en-us/library/Mt684959(v=VS.85).aspx) | 启用范围跟踪功能，用于更新序列号(USN)更改目标卷上的日志流，或修改已启用的范围跟踪参数。  Enables range tracking feature for update sequence number (USN) change journal stream on a target volume, or modifies already enabled range tracking parameters. |
| [**FSCTL\_WRITE\_USN\_CLOSE\_RECORD**](https://msdn.microsoft.com/en-us/library/Aa364816(v=VS.85).aspx) | 在输入文件的更新序列号(USN)更改日志流中生成一条记录。  Generates a record in the update sequence number (USN) change journal stream for the input file. |
| [**IOCTL\_VOLUME\_GET\_GPT\_ATTRIBUTES**](https://docs.microsoft.com/en-us/windows/desktop/api/WinIoCtl/ni-winioctl-ioctl_volume_get_gpt_attributes) | 检索卷的属性。  Retrieves the attributes for a volume. To perform this operation, call the [**DeviceIoControl**](https://msdn.microsoft.com/library/windows/desktop/aa363216) function with the following parameters. |
| [**IOCTL\_VOLUME\_GET\_VOLUME\_DISK\_EXTENTS**](https://docs.microsoft.com/en-us/windows/desktop/api/WinIoCtl/ni-winioctl-ioctl_volume_get_volume_disk_extents) | 检索一个或多个磁盘上指定卷的物理位置。  Retrieves the physical location of a specified volume on one or more disks. |
| [**IOCTL\_VOLUME\_IS\_CLUSTERED**](https://docs.microsoft.com/en-us/windows/desktop/api/WinIoCtl/ni-winioctl-ioctl_volume_is_clustered) | 确定指定的卷是否is clustered。  Determines whether the specified volume is clustered. |
| [**IOCTL\_VOLUME\_IS\_CSV**](https://docs.microsoft.com/zh-cn/windows/desktop/FileIO/ioctl-volume-is-csv) | 确定一个卷是否是CSV卷。  Determines whether a volume is a CSV volume. |
| [**IOCTL\_VOLUME\_OFFLINE**](https://docs.microsoft.com/en-us/windows/desktop/api/WinIoCtl/ni-winioctl-ioctl_volume_offline) | 使卷脱机。  Takes a volume offline. |
| [**IOCTL\_VOLUME\_ONLINE**](https://docs.microsoft.com/en-us/windows/desktop/api/WinIoCtl/ni-winioctl-ioctl_volume_online) | 卷在线。  Brings a volume online. |
|  |  |

**注意**：黄底红字标识的控制代码属于变更日志（[change journals](https://docs.microsoft.com/zh-cn/windows/desktop/FileIO/change-journals)）使用的控制代码。

从卷中取得日志通过[**FSCTL\_CREATE\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364558(v=VS.85).aspx)（在目标卷上创建更新序列号（USN）变更日志（Change Jounral），或修改现有更改日志）控制代码进行。API调用如下：

|  |
| --- |
| C++ |
| BOOL  WINAPI  DeviceIoControl((HANDLE)hDevice, // handle to volume  **FSCTL\_CREATE\_USN\_JOURNAL**, // dwIoControlCode  (LPVOID)lpInBuffer, // input buffer  (DWORD)nInBufferSize, // size of input buffer  NULL, // lpOutBuffer  0, // nOutBufferSize  (LPDWORD)lpBytesReturned, // number of bytes returned  (LPOVERLAPPED)lpOverlapped); // OVERLAPPED structure |

参考：<https://msdn.microsoft.com/en-us/library/Aa364558%28v=VS.85%29.aspx?f=255&MSPPError=-2147217396>

|  |  |
| --- | --- |
| hDevice | 要创建或修改的更改日志流的本地卷的句柄。要获得卷句柄，请调用CreateFile函数。  A handle to the local volume for the change journal stream that is to be created or modified.To retrieve a volume handle, call the [CreateFile](https://msdn.microsoft.com/en-us/library/aa363858(v=vs.85).aspx) function. |
| dwIoControlCode | 操作的控制代码。此处使用FSCTL\_CREATE\_USN\_JOURNAL。其作用是有日志已经存在则打开以供读取或修改，没有日志就创建一个新的供读取或修改。  The control code for the operation.Use **FSCTL\_CREATE\_USN\_JOURNAL** for this operation. |
| lpInBuffer | 一个指向CREATE\_USN\_JOURNAL\_DATA结构的输入缓冲区的指针。  A pointer to the input buffer, a [**CREATE\_USN\_JOURNAL\_DATA**](https://msdn.microsoft.com/en-us/library/aa363871(v=vs.85).aspx) structure. |
| nInBufferSize | 输入缓冲区的大小，以字节为单位。  The size of the input buffer, in bytes. |
| lpOutBuffer | 此处不用;设置为**NULL**。  Not used with this operation; set to **NULL**. |
| nOutBufferSize | 此处不用;设置为**0**。  Not used with this operation; set to zero. |
| lpBytesReturned | 指向一个变量的指针，该变量接收以字节为单位存储在输出缓冲区中的数据的大小。  如果lpOverlapped为空，则lpBytesReturned不能为空。即使一个操作不返回输出数据，并且lpOutBuffer为NULL,DeviceIoControl仍然要使用lpBytesReturned，即使返回的lpBytesReturned的值是没有意义的。  如果lpOverlapped不为空，则lpBytesReturned可以为空。如果该参数不为空，且操作返回数据，则在重叠（异步）操作完成之前，返回的lpBytesReturned是无意义的。要检索返回的字节数，请调用[**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx)。如果hDevice与**I/O完成端口**相关联，则可以通过调用[**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx)检索返回的字节数。  A pointer to a variable that receives the size of the data that is stored in the output buffer, in bytes.  If *lpOverlapped* is **NULL**, *lpBytesReturned* cannot be **NULL**. Even when an operation does not return output data and *lpOutBuffer* is **NULL**,[**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) uses *lpBytesReturned*. After such an operation, the value of *lpBytesReturned* is meaningless.  If *lpOverlapped* is not **NULL**, *lpBytesReturned* can be **NULL**. If this parameter is not **NULL** and the operation returns data, *lpBytesReturned* is meaningless until the overlapped operation is complete. To retrieve the number of bytes returned, call [**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx). If *hDevice* is associated with an I/O completion port, you can retrieve the number of bytes returned by calling [**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx). |
| lpOverlapped | 指向重叠结构（[**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) ）的指针。如果在没有指定FILE\_FLAG\_OVERLAPPED的情况下打开hDevice，则忽略lpOverlapped。  如果使用FILE\_FLAG\_OVERLAPPED标志打开hDevice，则该操作将作为重叠(异步)操作执行。在这种情况下，lpOverlapped必须指向一个有效的重叠结构，该结构包含一个事件对象的句柄。否则，函数将以不可预知的方式失败。  对于重叠操作，DeviceIoControl将立即返回，当操作完成时，事件对象将发出信号。否则，在操作完成或出现错误之前，函数不会返回。  A pointer to an [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure.  If *hDevice* is opened without specifying **FILE\_FLAG\_OVERLAPPED**, *lpOverlapped* is ignored.  If *hDevice* is opened with the **FILE\_FLAG\_OVERLAPPED** flag, the operation is performed as an overlapped (asynchronous) operation. In this case,*lpOverlapped* must point to a valid [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure that contains a handle to an event object. Otherwise, the function fails in unpredictable ways.  For overlapped operations, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns immediately, and the event object is signaled when the operation is complete. Otherwise, the function does not return until the operation is complete or an error occurs. |
|  |  |

**CREATE\_USN\_JOURNAL\_DATA structure**描述更新序列号(USN)变更改日志（change journal）的信息。

|  |  |
| --- | --- |
| C++ | |
| typedef struct {  DWORDLONG MaximumSize;  DWORDLONG AllocationDelta;  } CREATE\_USN\_JOURNAL\_DATA, \*PCREATE\_USN\_JOURNAL\_DATA; | |
| MaximumSize | NTFS文件系统为更改日志分配的目标最大大小(以字节为单位)。更改日志可以比这个值增长得更大，但是在下一个NTFS文件系统检查点将其截断为小于这个值。  The target maximum size that the NTFS file system allocates for the change journal, in bytes. The change journal can grow larger than this value, but it is then truncated at the next NTFS file system checkpoint to less than this value. |
| AllocationDelta | 添加到更改日志末尾并从开头删除的内存分配大小，以字节为单位。  更改日志可以增长到MaximumSize 和AllocationDelta值之和以上，然后再进行修剪。  The size of memory allocation that is added to the end and removed from the beginning of the change journal, in bytes.  The change journal can grow to more than the sum of the values of  MaximumSize and AllocationDelta before being trimmed. |
|  |  |

参考：<https://docs.microsoft.com/zh-cn/windows/desktop/api/winioctl/ns-winioctl-create_usn_journal_data>

**Return code**

|  |  |
| --- | --- |
| Value | Meaning |
| **ERROR\_INVALID\_FUNCTION** | 指定的卷不支持更改日志。  The specified volume does not support change journals. |
| **ERROR\_INVALID\_PARAMETER** | 一个或多个参数无效，例如，如果提供的句柄不是卷句柄，DeviceIoControl将返回此错误代码。  One or more parameters is invalid, for example, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns this error code if the handle supplied is not a volume handle. |
| **ERROR\_JOURNAL\_DELETE\_IN\_PROGRESS** | 尝试在日志删除过程中读取、创建、删除或修改日志，或者尝试在日志删除过程中写入USN记录。  An attempt is made to read from, create, delete, or modify the journal while a journal deletion is in process, or an attempt is made to write a USN record while a journal deletion is in process. |
|  |  |

|  |
| --- |
| 备注 |
| 如果操作成功完成，DeviceIoControl返回一个非零值。  如果操作失败或挂起，DeviceIoControl返回零。要获取错误信息，请调用GetLastError。可能的返回值参见上表（**Return code**）。  可以使用FSCTL\_CREATE\_USN\_JOURNAL为卷创建一个新的变更日志。创建之后，NTFS文件系统为该卷维护一个变更改日志（$UsnJrnl）。还可以使用FSCTL\_CREATE\_USN\_JOURNAL修改现有的变更日志。  如果变更日志已经存在，FSCTL\_CREATE\_USN\_JOURNAL将其设置为CREATE\_USN\_JOURNAL\_DATA结构中提供的特征。变更日志最终会变得更大，或者被调整到CREATE\_USN\_JOURNAL\_DATA的MaximumSize限制。 |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  DWORD lpBytesReturned = 0;  CREATE\_USN\_JOURNAL\_DATA cujd;  cujd.MaximumSize = 0; //0表示使用默认值  cujd.AllocationDelta = 0; //0表示使用默认值  BOOL bOk = DeviceIoControl(hVol,  FSCTL\_CREATE\_USN\_JOURNAL,  &cujd,  sizeof(cujd),  NULL,  0,  &lpBytesReturned,  NULL);  if (bOk)  {  printf("从卷中取得日志成功");  }  else  {  printf("初始化USN日志文件失败 error:%d\n", GetLastError());  } |

四、读取日志的信息

从卷中读取日志通过[**FSCTL\_QUERY\_USN\_JOURNAL**](https://msdn.microsoft.com/en-us/library/Aa364558(v=VS.85).aspx)（查询关于当前更新序列号（USN）更改日志（Change Jounral）、其记录和容量的信息）控制代码进行。API调用如下：

|  |
| --- |
| C++ |
| BOOL  WINAPI  DeviceIoControl((HANDLE)hDevice, // handle to volume  **FSCTL\_QUERY\_USN\_JOURNAL**, // dwIoControlCode  (LPVOID)lpInBuffer, // input buffer  (DWORD)nInBufferSize, // size of input buffer  NULL, // lpOutBuffer  0, // nOutBufferSize  (LPDWORD)lpBytesReturned, // number of bytes returned  (LPOVERLAPPED)lpOverlapped); // OVERLAPPED structure |

参考：<https://msdn.microsoft.com/en-us/library/Aa364583(v=VS.85).aspx>

|  |  |
| --- | --- |
| hDevice | 要创建或修改的更改日志流的本地卷的句柄。要获得卷句柄，请调用CreateFile函数。  A handle to the local volume for the change journal stream that is to be created or modified.To retrieve a volume handle, call the [CreateFile](https://msdn.microsoft.com/en-us/library/aa363858(v=vs.85).aspx) function. |
| dwIoControlCode | 操作的控制代码。此处使用FSCTL\_QUERY\_USN\_JOURNAL。  The control code for the operation.Use **FSCTL\_QUERY\_USN\_JOURNAL** for this operation. |
| lpInBuffer | 此处不用;设置为**NULL**。  Not used with this operation; set to **NULL**. |
| nInBufferSize | 此处不用;设置为**0**。  Not used with this operation; set to zero. |
| lpOutBuffer | 一个指向输出缓冲区的指针，它接收一个USN\_JOURNAL\_DATA\_V0或USN\_JOURNAL\_DATA\_V1或USN\_JOURNAL\_DATA\_V2结构，该结构包含关于当前更改日志、记录和容量的信息。  A pointer to the output buffer that  receives a [USN\_JOURNAL\_DATA\_V0](https://msdn.microsoft.com/en-us/library/aa365721(v=vs.85).aspx) or [USN\_JOURNAL\_DATA\_V1](https://msdn.microsoft.com/en-us/library/hh802707(v=vs.85).aspx) or [USN\_JOURNAL\_DATA\_V2](https://msdn.microsoft.com/en-us/library/mt684960(v=vs.85).aspx) structure  that contains information about the current change journal, its records, and its capacity. |
| nOutBufferSize | 输出缓冲区的大小，以字节为单位。如果传递的大小至少是sizeof(USN\_JOURNAL\_DATA\_V0)，但小于sizeof(USN\_JOURNAL\_DATA\_V1)，则输出结构是USN\_JOURNAL\_DATA\_V0结构。  如果传递的大小至少是sizeof(USN\_JOURNAL\_DATA\_V1)，  但小于sizeof(USN\_JOURNAL\_DATA\_V2)，则输出缓冲区是USN\_JOURNAL\_DATA\_V1结构。如果输出缓冲区的大小(USN\_JOURNAL\_DATA\_V2)或更大，则输出缓冲区是USN\_JOURNAL\_DATA\_V2结构。  Windows Server 2008 R2、Windows 7、Windows Server 2008、Windows Vista、Windows Server 2003和Windows XP: USN\_JOURNAL\_DATA\_V1在Windows 8和Windows Server 2012之前不受支持。在Windows 8.1和Windows Server 2012 R2之前不支持USN\_JOURNAL\_DATA\_V2。  The size of the output buffer, in bytes. If the size passed is at least sizeof(USN\_JOURNAL\_DATA\_V0) but less than sizeof(USN\_JOURNAL\_DATA\_V1), the output structure is a [**USN\_JOURNAL\_DATA\_V0**](https://msdn.microsoft.com/en-us/library/aa365721(v=vs.85).aspx) structure. If the size passed is at least sizeof(USN\_JOURNAL\_DATA\_V1) but less thansizeof(USN\_JOURNAL\_DATA\_V2), the output buffer is a [**USN\_JOURNAL\_DATA\_V1**](https://msdn.microsoft.com/en-us/library/hh802707(v=vs.85).aspx) structure. If the output buffer is sizeof(USN\_JOURNAL\_DATA\_V2)or larger, the output buffer is a [**USN\_JOURNAL\_DATA\_V2**](https://msdn.microsoft.com/en-us/library/mt684960(v=vs.85).aspx) structure.  **Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP:**[**USN\_JOURNAL\_DATA\_V1**](https://msdn.microsoft.com/en-us/library/hh802707(v=vs.85).aspx) is not supported before Windows 8 and Windows Server 2012. [**USN\_JOURNAL\_DATA\_V2**](https://msdn.microsoft.com/en-us/library/mt684960(v=vs.85).aspx) is not supported before Windows 8.1 and Windows Server 2012 R2. |
| lpBytesReturned | 指向一个变量的指针，该变量接收以字节为单位存储在输出缓冲区中的数据的大小。  如果lpOverlapped为空，则lpBytesReturned不能为空。即使一个操作不返回输出数据，并且lpOutBuffer为NULL,DeviceIoControl仍然要使用lpBytesReturned，即使返回的lpBytesReturned的值是没有意义的。  如果lpOverlapped不为空，则lpBytesReturned可以为空。如果该参数不为空，且操作返回数据，则在重叠（异步）操作完成之前，返回的lpBytesReturned是无意义的。要检索返回的字节数，请调用[**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx)。如果hDevice与**I/O完成端口**相关联，则可以通过调用[**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx)检索返回的字节数。  A pointer to a variable that receives the size of the data that is stored in the output buffer, in bytes.  If *lpOverlapped* is **NULL**, *lpBytesReturned* cannot be **NULL**. Even when an operation does not return output data and *lpOutBuffer* is **NULL**,[**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) uses *lpBytesReturned*. After such an operation, the value of *lpBytesReturned* is meaningless.  If *lpOverlapped* is not **NULL**, *lpBytesReturned* can be **NULL**. If this parameter is not **NULL** and the operation returns data, *lpBytesReturned* is meaningless until the overlapped operation is complete. To retrieve the number of bytes returned, call [**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx). If *hDevice* is associated with an I/O completion port, you can retrieve the number of bytes returned by calling [**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx). |
| lpOverlapped | 指向重叠结构（[**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) ）的指针。如果在没有指定FILE\_FLAG\_OVERLAPPED的情况下打开hDevice，则忽略lpOverlapped。  如果使用FILE\_FLAG\_OVERLAPPED标志打开hDevice，则该操作将作为重叠(异步)操作执行。在这种情况下，lpOverlapped必须指向一个有效的重叠结构，该结构包含一个事件对象的句柄。否则，函数将以不可预知的方式失败。  对于重叠操作，DeviceIoControl将立即返回，当操作完成时，事件对象将发出信号。否则，在操作完成或出现错误之前，函数不会返回。  A pointer to an [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure.  If *hDevice* is opened without specifying **FILE\_FLAG\_OVERLAPPED**, *lpOverlapped* is ignored.  If *hDevice* is opened with the **FILE\_FLAG\_OVERLAPPED** flag, the operation is performed as an overlapped (asynchronous) operation. In this case,*lpOverlapped* must point to a valid [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure that contains a handle to an event object. Otherwise, the function fails in unpredictable ways.  For overlapped operations, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns immediately, and the event object is signaled when the operation is complete. Otherwise, the function does not return until the operation is complete or an error occurs. |
|  |  |

[**USN\_JOURNAL\_DATA\_V**](https://msdn.microsoft.com/en-us/library/aa365721(v=vs.85).aspx)x **structure** 记录更新序列号（USN）变更日志（Change Journal）的容量等信息，它是日志记录的入口。这个结构是FSCTL\_QUERY\_USN\_JOURNAL控制代码的输出缓冲区（lpOutBuffer）。在Windows 8和Windows Server 2012之前，这个结构被命名为USN\_JOURNAL\_DATA且需要使用旧的SDK和编译器来编译。

|  |  |
| --- | --- |
| C++ | |
| typedef struct {  DWORDLONG UsnJournalID; // V0 **supported** >=Windows XP [desktop apps only] 下同  USN FirstUsn; // V0  USN NextUsn; // V0  USN LowestValidUsn; // V0  USN MaxUsn; // V0  DWORDLONG MaximumSize; // V0  DWORDLONG AllocationDelta; // V0  WORD MinSupportedMajorVersion; //V1 **supported** >=Windows 8 [desktop apps only]  WORD MaxSupportedMajorVersion; //V1 **supported** >= Windows 8 [desktop apps only]  DWORD Flags; // V2 **supported** >=Windows 8.1 [desktop apps only]  DWORDLONG RangeTrackChunkSize; // V2 **supported** >=Windows 8.1 [desktop apps only]  LONGLONG RangeTrackFileSizeThreshold; // V2 **supported** >=Windows 8.1 [desktop apps only]  } USN\_JOURNAL\_DATA\_Vx, \*PUSN\_JOURNAL\_DATA\_Vx;  #if (NTDDI\_VERSION >= NTDDI\_WIN8) //Win8及以后版本自动选用V1版了  typedef USN\_JOURNAL\_DATA\_V1 USN\_JOURNAL\_DATA, \*PUSN\_JOURNAL\_DATA;  #else  typedef USN\_JOURNAL\_DATA\_V0 USN\_JOURNAL\_DATA, \*PUSN\_JOURNAL\_DATA;  #endif | |
| UsnJournalID | 当前日志标识符。日志在创建时被分配一个新的标识符，并且可以在其存在的过程中使用一个新的标识符进行标记。NTFS文件系统使用此标识符进行完整性检查。  The current journal identifier. A journal is assigned a new identifier on creation and can be stamped with a new identifier in the course of its existence. The NTFS file system uses this identifier for an integrity check. |
| FirstUsn | 可以从日志中读取的第一个记录的号码。  The number of first record that can be read from the journal. |
| NextUsn | 被写入日志的下一个记录号。  The number of next record to be written to the journal. |
| LowestValidUsn | 为这个日志实例写入日志的第一个记录。枚举卷上的文件或目录可以返回一个低于此值的USN(换句话说，FirstUsn成员值小于LowestValidUsn成员值)。如果是，则自上次编写USN以来，日志已被加盖新标识符。在本例中，LowestValidUsn可能表示日志中的不连续，其中对卷上的某些或所有文件或目录的更改可能发生在更改日志中没有记录的更改中。  The first record that was written into the journal for this journal instance. Enumerating the files or directories on a volume can return a USN lower than this value (in other words, a FirstUsn member value less than the LowestValidUsn member value). If it does, the journal has been stamped with a new identifier since the last USN was written. In this case, LowestValidUsn may indicate a discontinuity in the journal, in which changes to some or all files or directories on the volume may have occurred that are not recorded in the change journal. |
| MaxUsn | 变更日志支持的最大USN。当NextUsn的值接近这个值时，管理员必须删除更改日志。  The largest USN that the change journal supports. An administrator must delete the change journal as the value of NextUsn approaches this value. |
| MaximumSize | 更改日志的目标最大大小，以字节为单位。更改日志可以比这个值增长得更大，但是在下一个NTFS文件系统检查点将其截断为小于这个值。  The target maximum size for the change journal, in bytes. The change journal can grow larger than this value, but it is then truncated at the next NTFS file system checkpoint to less than this value. |
| AllocationDelta | 每次分配或释放内存时，添加到更改日志末尾并从更改日志开始删除的磁盘内存字节数。换句话说，分配和重新分配以这种大小的单元进行。对于这个成员，分配簇大小的整数倍是一个合理的值。  The number of bytes of disk memory added to the end and removed from the beginning of the change journal each time memory is allocated or deallocated. In other words, allocation and deallocation take place in units of this size. An integer multiple of a cluster size is a reasonable value for this member. |
| MinSupportedMajorVersion | 返回文件系统支持的USN变更日志的最小支持版本。  Indicates the minimum supported version of the USN change journal supported by the filesystem. |
| MaxSupportedMajorVersion | 返回文件系统支持的USN变更日志的最大支持版本。  Indicates the maximum supported version of the USN change journal supported by the filesystem. |
| Flags | 是否开启范围跟踪。下面是Flags成员的可能值。  0x00000000：卷标范围跟踪未打开。Range tracking is not turned on for the volume.  0x00000001(**FLAG\_USN\_TRACK\_MODIFIED\_RANGES\_ENABLE**)：卷标范围跟踪已打开。Range tracking is turned on for the volume. |
| RangeTrackChunkSize | 跟踪范围粒度。仅当您还将Flags成员设置为FLAG\_USN\_TRACK\_MODIFIED\_RANGES\_ENABLE时才有效。  The granularity of tracked ranges. Valid only when you also set  the **Flags** member to **FLAG\_USN\_TRACK\_MODIFIED\_RANGES\_ENABLE**. |
| RangeTrackFileSizeThreshold | 文件大小阈值，用于开始范围跟踪相等或较大尺寸的文件。仅当您将Flags成员设置为FLAG\_USN\_TRACK\_MODIFIED\_RANGES\_ENABLE时才有效。  File size threshold to start tracking range for files with equal or larger size. Valid only when you also set the **Flags** member to **FLAG\_USN\_TRACK\_MODIFIED\_RANGES\_ENABLE**. |
|  |  |

参考：<https://docs.microsoft.com/zh-cn/windows/desktop/api/winioctl/ns-winioctl-usn_journal_data_v0>

**Return code**

|  |  |
| --- | --- |
| Value | Meaning |
| **ERROR\_INVALID\_FUNCTION** | 指定的卷不支持变更日志。  The specified volume does not support change journals. |
| **ERROR\_INVALID\_PARAMETER** | 一个或多个参数无效，例如，如果提供的句柄不是卷句柄，DeviceIoControl将返回此错误代码。  One or more parameters is invalid, for example, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns this error code if the handle supplied is not a volume handle. |
|  |  |

|  |
| --- |
| 备注 |
| 如果操作成功完成，DeviceIoControl返回一个非零值。  如果操作失败或挂起，DeviceIoControl返回零。要获取错误信息，请调用GetLastError。可能的返回值参见上表（**Return code**）。  可以使用FSCTL\_CREATE\_USN\_JOURNAL为卷创建一个新的变更日志。创建之后，NTFS文件系统为该卷维护一个变更日志（$UsnJrnl）。还可以使用FSCTL\_CREATE\_USN\_JOURNAL修改现有的变更日志。  如果变更日志已经存在，FSCTL\_CREATE\_USN\_JOURNAL将其设置为CREATE\_USN\_JOURNAL\_DATA结构中提供的特征。变更日志最终会变得更大，或者被调整到CREATE\_USN\_JOURNAL\_DATA的MaximumSize限制。 |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  DWORD dwBytesReturned = 0;  USN\_JOURNAL\_DATA ujd;  BOOL bOk = DeviceIoControl(hVol,  FSCTL\_QUERY\_USN\_JOURNAL,  NULL,  0,  &ujd,  sizeof(USN\_JOURNAL\_DATA),  &dwBytesReturned,  NULL);  if (bOk)  {  printf("读取日志的信息成功");  getBasicInfoSuccess = true;  }  else  {  printf("读取日志的信息失败 error:%d\n", GetLastError());  } |

五、枚举所有的数据

从卷中读取日志通过[**FSCTL\_ENUM\_USN\_DATA**](https://msdn.microsoft.com/en-us/library/Aa364563(v=VS.85).aspx)（枚举两个指定边界之间的更新序列号(USN)数据，以获取主文件表(MFT)记录。）控制代码进行。API调用如下：

|  |
| --- |
| C++ |
| BOOL  WINAPI  DeviceIoControl((HANDLE)hDevice, // handle to volume  **FSCTL\_ENUM\_USN\_DATA**, // dwIoControlCode  (LPVOID)lpInBuffer, // input buffer  (DWORD)nInBufferSize, // size of input buffer  (LPVOID)lpOutBuffer, // lpOutBuffer  (DWORD)nOutBufferSize, // nOutBufferSize  (LPDWORD)lpBytesReturned, // number of bytes returned  (LPOVERLAPPED)lpOverlapped); // OVERLAPPED structure |

参考：<https://msdn.microsoft.com/en-us/library/Aa364563%28v=VS.85%29.aspx?f=255&MSPPError=-2147217396>

|  |  |
| --- | --- |
| hDevice | 要检索数据的卷的句柄。  A handle to the local volume from which data is to be retrieved.  To retrieve a volume handle, call the [CreateFile](https://msdn.microsoft.com/en-us/library/aa363858(v=vs.85).aspx) function. |
| dwIoControlCode | 操作的控制代码。此处使用FSCTL\_QUERY\_USN\_JOURNAL。  The control code for the operation.Use **FSCTL\_QUERY\_USN\_JOURNAL** for this operation. |
| lpInBuffer | 指向输入缓冲区、MFT\_ENUM\_DATA\_V0或MFT\_ENUM\_DATA\_V1结构的指针。此结构指定列出MFT条目之间的边界。  Windows Server 2008 R2、Windows 7、Windows Server 2008、Windows Vista、Windows Server 2003和Windows XP：在Windows Server 2012之前不支持MFT\_ENUM\_DATA\_V1结构  A pointer to the input buffer, an [MFT\_ENUM\_DATA\_V0](https://msdn.microsoft.com/en-us/library/aa365232(v=vs.85).aspx) or [MFT\_ENUM\_DATA\_V1](https://msdn.microsoft.com/en-us/library/hh802704(v=vs.85).aspx) structure.  This structure specifies the boundaries between which MFT entries are listed.  Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP:  [MFT\_ENUM\_DATA\_V1](https://msdn.microsoft.com/en-us/library/hh802704(v=vs.85).aspx) structures are not supported before Windows Server 2012. |
| nInBufferSize | 输入缓冲区的大小，以字节为单位。传递的大小决定系统是否在MFT\_ENUM\_DATA\_V1结构的末尾查找其他成员。  The size of the input buffer, in bytes. The size passed determines whether the system looks for the additional members at the end of the [**MFT\_ENUM\_DATA\_V1**](https://msdn.microsoft.com/en-us/library/hh802704(v=vs.85).aspx) structure. |
| lpOutBuffer | 一个指向输出缓冲区的指针，它接收一个USN，然后是零个或多个USN\_RECORD\_V2或USN\_RECORD\_V3结构。USN是一个DWORDLONG值，表示缓冲区中最后一条记录之后的USN。它可以在随后的toFSCTL\_ENUM\_USN\_DATA调用中使用，以获得当前输出缓冲区中的记录之后的下一组记录。有关在输出缓冲区中导航数据的更多信息，请参见遍历更改日志记录的缓冲区。Windows Server 2008 R2、Windows 7、Windows Server 2008、Windows Vista、Windows Server 2003和Windows XP:在Windows Server 2012之前不支持 [USN\_RECORD\_V3](https://msdn.microsoft.com/en-us/library/hh802708(v=vs.85).aspx)结构。  A pointer to the output buffer that receives a USN followed by zero  or more [USN\_RECORD\_V2](https://msdn.microsoft.com/en-us/library/aa365722(v=vs.85).aspx) or [USN\_RECORD\_V3](https://msdn.microsoft.com/en-us/library/hh802708(v=vs.85).aspx) structures.  The USN is a DWORDLONG value that represents the USN following the last record in the buffer. It can be used in a subsequent call toFSCTL\_ENUM\_USN\_DATA to get the next set of records following those in the current output buffer.  For more information about navigating the data in the output buffer, see [Walking a Buffer of Change Journal Records](https://msdn.microsoft.com/en-us/library/aa365736(v=vs.85).aspx).  Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP:  [USN\_RECORD\_V3](https://msdn.microsoft.com/en-us/library/hh802708(v=vs.85).aspx)structures are not supported before Windows Server 2012. |
| nOutBufferSize | 输出缓冲区的大小，以字节为单位。  The size of the output buffer, in bytes. |
| lpBytesReturned | 指向一个变量的指针，该变量接收以字节为单位存储在输出缓冲区中的数据的大小。  如果lpOverlapped为空，则lpBytesReturned不能为空。即使一个操作不返回输出数据，并且lpOutBuffer为NULL,DeviceIoControl仍然要使用lpBytesReturned，即使返回的lpBytesReturned的值是没有意义的。  如果lpOverlapped不为空，则lpBytesReturned可以为空。如果该参数不为空，且操作返回数据，则在重叠（异步）操作完成之前，返回的lpBytesReturned是无意义的。要检索返回的字节数，请调用[**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx)。如果hDevice与**I/O完成端口**相关联，则可以通过调用[**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx)检索返回的字节数。  A pointer to a variable that receives the size of the data that is stored in the output buffer, in bytes.  If *lpOverlapped* is **NULL**, *lpBytesReturned* cannot be **NULL**. Even when an operation does not return output data and *lpOutBuffer* is **NULL**,[**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) uses *lpBytesReturned*. After such an operation, the value of *lpBytesReturned* is meaningless.  If *lpOverlapped* is not **NULL**, *lpBytesReturned* can be **NULL**. If this parameter is not **NULL** and the operation returns data, *lpBytesReturned* is meaningless until the overlapped operation is complete. To retrieve the number of bytes returned, call [**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx). If *hDevice* is associated with an I/O completion port, you can retrieve the number of bytes returned by calling [**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx). |
| lpOverlapped | 指向重叠结构（[**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) ）的指针。如果在没有指定FILE\_FLAG\_OVERLAPPED的情况下打开hDevice，则忽略lpOverlapped。  如果使用FILE\_FLAG\_OVERLAPPED标志打开hDevice，则该操作将作为重叠(异步)操作执行。在这种情况下，lpOverlapped必须指向一个有效的重叠结构，该结构包含一个事件对象的句柄。否则，函数将以不可预知的方式失败。  对于重叠操作，DeviceIoControl将立即返回，当操作完成时，事件对象将发出信号。否则，在操作完成或出现错误之前，函数不会返回。  A pointer to an [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure.  If *hDevice* is opened without specifying **FILE\_FLAG\_OVERLAPPED**, *lpOverlapped* is ignored.  If *hDevice* is opened with the **FILE\_FLAG\_OVERLAPPED** flag, the operation is performed as an overlapped (asynchronous) operation. In this case,*lpOverlapped* must point to a valid [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure that contains a handle to an event object. Otherwise, the function fails in unpredictable ways.  For overlapped operations, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns immediately, and the event object is signaled when the operation is complete. Otherwise, the function does not return until the operation is complete or an error occurs. |
|  |  |

**MFT\_ENUM\_DATA\_V**x **structure** 记录更新序列号（USN）变更日志（Change Journal）枚举的边界和起始位置的信息。这个结构是FSCTL\_ENUM\_USN\_DATA控制代码的输入缓冲区（lpInBuffer）。在Windows 8和Windows Server 2012之前，这个结构被命名为USN\_ENUM\_DATA且需要使用旧的SDK和编译器来编译。

|  |  |
| --- | --- |
| C++ | |
| typedef struct {  DWORDLONG StartFileReferenceNumber;  USN LowUsn;  USN HighUsn;  WORD MinMajorVersion; //V1 **supported** >= Windows Server 2012 [desktop apps only]  WORD MaxMajorVersion; //V1 **supported** >= Windows Server 2012 [desktop apps only]  } MFT\_ENUM\_DATA\_Vx, \*PMFT\_ENUM\_DATA\_Vx; | |
| StartFileReferenceNumber | 要在当前卷上开始枚举的文件中的序号位置。  枚举过程中对FSCTL\_ENUM\_USN\_DATA的第一个调用必须将StartFileReferenceNumber成员设置为(DWORDLONG)0。每个对FSCTL\_ENUM\_USN\_DATA的调用都检索后续调用的起始点，作为输出缓冲区中的第一个条目。后续调用必须将StartFileReferenceNumber设置为这个值。  The ordinal position within the files on the current volume at which the enumeration is to begin.  The first call to [FSCTL\_ENUM\_USN\_DATA](https://msdn.microsoft.com/44d20401-a2ed-4756-9fda-878a24eab7c3) during an enumeration must have the StartFileReferenceNumber member set to (DWORDLONG)0. Each call to FSCTL\_ENUM\_USN\_DATA retrieves the starting point for the subsequent call as the first entry in the output buffer. Subsequent calls must be made with StartFileReferenceNumber set to this value. |
| LowUsn | 用于筛选返回的记录的USN值范围的下界。只返回上次更改日志USN介于或等于LowUsn和HighUsn成员值之间的记录。  The lower boundary of the range of USN values used to filter which records are returned. Only records whose last change journal USN is between or equal to the **LowUsn** and **HighUsn** member values are returned. |
| HighUsn | 用于筛选返回哪些文件的USN值范围的上边界。  The upper boundary of the range of USN values used to filter which files are returned. |
| MinMajorVersion | 指示USN更改日志所支持的最小主版本。  Indicates the minimum supported major version for the USN change journal. |
| MaxMajorVersion | 指示USN更改日志所支持的最大主版本。  Indicates the maximum supported major version for the USN change journal.  2：从FSCTL\_ENUM\_USN\_DATA控制代码返回的数据将包含USN\_RECORD\_V2结构  The data returned from the [FSCTL\_ENUM\_USN\_DATA](https://msdn.microsoft.com/44d20401-a2ed-4756-9fda-878a24eab7c3) control code will contain [USN\_RECORD\_V2](https://msdn.microsoft.com/1747453d-fd18-4853-a953-47131f3067ae) structures. |
|  | 3：从FSCTL\_ENUM\_USN\_DATA控制代码返回的数据将包含USN\_RECORD\_V2  或USN\_RECORD\_V3结构。  The data returned from the [FSCTL\_ENUM\_USN\_DATA](https://msdn.microsoft.com/44d20401-a2ed-4756-9fda-878a24eab7c3) control code  will contain [USN\_RECORD\_V2](https://msdn.microsoft.com/1747453d-fd18-4853-a953-47131f3067ae) or [USN\_RECORD\_V3](https://msdn.microsoft.com/6d95c5d1-6c6b-498f-a00d-eaa540e8b15b) structures. |
|  |  |

参考：<https://docs.microsoft.com/zh-cn/windows/desktop/api/winioctl/ns-winioctl-mft_enum_data_v1>

**USN\_RECORD\_V**x **structure** 记录更新序列号（USN）变更日志（Change Journal）记录信息。这个结构是FSCTL\_ENUM\_USN\_DATA控制代码的输出缓冲区（lpInBuffer）。在Windows 8和Windows Server 2012之前，这个结构被命名为USN\_ENUM\_DATA且需要使用旧的SDK和编译器来编译。USN\_RECORD在前面已经介绍不再重复。

**Return code**

|  |  |
| --- | --- |
| Value | Meaning |
| **ERROR\_INVALID\_FUNCTION** | 指定的卷不支持变更日志。  The specified volume does not support change journals. |
| **ERROR\_INVALID\_PARAMETER** | 一个或多个参数无效，例如，如果提供的句柄不是卷句柄，DeviceIoControl将返回此错误代码。  One or more parameters is invalid, for example, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns this error code if the handle supplied is not a volume handle. |
|  |  |

|  |
| --- |
| 备注 |
| 如果操作成功完成，DeviceIoControl返回一个非零值。  如果操作失败或挂起，DeviceIoControl返回零。要获取错误信息，请调用GetLastError。可能的返回值参见上表（**Return code**）。 |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  //在第一次调用时，将起点MFT\_ENUM\_DATA结构的StartFileReferenceNumber成员设置为(DWORDLONG)0。每//个对FSCTL\_ENUM\_USN\_DATA的调用都检索后续调用的起始点，作为输出缓冲区中的第一个条目。  // MSDN: On the first call, set the starting point, the StartFileReferenceNumber member of the MFT\_ENUM\_DATA //structure, to (DWORDLONG)0.  // Each call to FSCTL\_ENUM\_USN\_DATA retrieves the starting point for the subsequent call as the first entry in the output //buffer.  MFT\_ENUM\_DATA med;  med.StartFileReferenceNumber = 0;  med.LowUsn = 0;//UsnInfo.FirstUsn; 这里经测试发现，如果用FirstUsn有时候不正确，导致获取到不完整的数据.  med.HighUsn = UsnInfo.NextUsn;  CHAR buffer[BUF\_LEN]; // 用于储存记录的缓冲,尽量足够地大  DWORD usnDataSize;  PUSN\_RECORD UsnRecord;  while (0 != DeviceIoControl(hVol,  FSCTL\_ENUM\_USN\_DATA,  &med,  sizeof(med),  buffer,  BUF\_LEN,  &usnDataSize,  NULL))  {  // 返回读取到的字节数  DWORD dwRetBytes = usnDataSize - sizeof(USN);  //返回第一个USN记录，后面跟着零个或多个更改日志记录，每个日志记录位于USN\_RECORD结构中。  // MSDN: return a USN followed by zero or more change journal records, each in a USN\_RECORD structure.  UsnRecord = (PUSN\_RECORD)(((PCHAR)buffer) + sizeof(USN));  while (dwRetBytes>0)  {  // 获取信息  const int strLen = UsnRecord->FileNameLength;  char fileName[MAX\_PATH] = { 0 };  WideCharToMultiByte(CP\_OEMCP, NULL, UsnRecord->FileName, strLen / 2, fileName, strLen, NULL, FALSE);    // 获取下一个记录  DWORD recordLen = UsnRecord->RecordLength;  dwRetBytes -= recordLen;  UsnRecord = (PUSN\_RECORD)(((PCHAR)UsnRecord) + recordLen);  }  //获取下一页数据  // The USN returned as the first item in the output buffer is the USN of the next record number to be retrieved.  // Use this value to continue reading records from the end boundary forward.  med.StartFileReferenceNumber = \*(USN \*)&buffer;  } |

六、关闭日志

关闭日志时可以删除卷上的更新序列号（USN）变更日志（Change Journal），但没有多大意义。

|  |
| --- |
| C++ |
| BOOL  WINAPI  DeviceIoControl((HANDLE)hDevice, // handle to volume  FSCTL\_DELETE\_USN\_JOURNAL, // dwIoControlCode  (LPVOID)lpInBuffer, // input buffer  (DWORD)nInBufferSize, // size of input buffer  NULL, // lpOutBuffer  0, // nOutBufferSize  (LPDWORD)lpBytesReturned, // number of bytes returned  (LPOVERLAPPED)lpOverlapped); // OVERLAPPED structure |

参考：<https://msdn.microsoft.com/en-us/library/Aa364561(v=VS.85).aspx>

|  |  |
| --- | --- |
| hDevice | 要从中删除变更日志的本地卷的句柄。要取得卷句柄，请调用CreateFile函数。  A handle to the local volume from which the change journal is to be deleted.  To retrieve a volume handle, call the [**CreateFile**](https://msdn.microsoft.com/en-us/library/aa363858(v=vs.85).aspx) function. |
| dwIoControlCode | 操作的控制代码。此处使用FSCTL\_DELETE\_USN\_JOURNAL。  The control code for the operation.Use **FSCTL\_DELETE\_USN\_JOURNAL** for this operation. |
| lpInBuffer | 一个指向DELETE\_USN\_JOURNAL\_DATA结构的输入缓冲区的指针。  A pointer to the input buffer, a [**DELETE\_USN\_JOURNAL\_DATA**](https://msdn.microsoft.com/en-us/library/aa363928(v=vs.85).aspx) structure. |
| nInBufferSize | 输入缓冲区的大小，以字节为单位。  The size of the input buffer, in bytes. |
| lpOutBuffer | 不用于此操作;设置为**NULL**。  Not used with this operation; set to **NULL**. |
| nOutBufferSize | 不用于此操作;设置为**0**。  Not used with this operation; set to zero. |
| lpBytesReturned | 指向一个变量的指针，该变量接收以字节为单位存储在输出缓冲区中的数据的大小。  如果lpOverlapped为空，则lpBytesReturned不能为空。即使一个操作不返回输出数据，并且lpOutBuffer为NULL,DeviceIoControl仍然要使用lpBytesReturned，即使返回的lpBytesReturned的值是没有意义的。  如果lpOverlapped不为空，则lpBytesReturned可以为空。如果该参数不为空，且操作返回数据，则在重叠（异步）操作完成之前，返回的lpBytesReturned是无意义的。要检索返回的字节数，请调用[**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx)。如果hDevice与**I/O完成端口**相关联，则可以通过调用[**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx)检索返回的字节数。  A pointer to a variable that receives the size of the data that is stored in the output buffer, in bytes.  If *lpOverlapped* is **NULL**, *lpBytesReturned* cannot be **NULL**. Even when an operation does not return output data and *lpOutBuffer* is **NULL**,[**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) uses *lpBytesReturned*. After such an operation, the value of *lpBytesReturned* is meaningless.  If *lpOverlapped* is not **NULL**, *lpBytesReturned* can be **NULL**. If this parameter is not **NULL** and the operation returns data, *lpBytesReturned* is meaningless until the overlapped operation is complete. To retrieve the number of bytes returned, call [**GetOverlappedResult**](https://msdn.microsoft.com/en-us/library/ms683209(v=vs.85).aspx). If *hDevice* is associated with an I/O completion port, you can retrieve the number of bytes returned by calling [**GetQueuedCompletionStatus**](https://msdn.microsoft.com/en-us/library/aa364986(v=vs.85).aspx). |
| lpOverlapped | 指向重叠结构（[**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) ）的指针。如果在没有指定FILE\_FLAG\_OVERLAPPED的情况下打开hDevice，则忽略lpOverlapped。  如果使用FILE\_FLAG\_OVERLAPPED标志打开hDevice，则该操作将作为重叠(异步)操作执行。在这种情况下，lpOverlapped必须指向一个有效的重叠结构，该结构包含一个事件对象的句柄。否则，函数将以不可预知的方式失败。  对于重叠操作，DeviceIoControl将立即返回，当操作完成时，事件对象将发出信号。否则，在操作完成或出现错误之前，函数不会返回。  A pointer to an [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure.  If *hDevice* is opened without specifying **FILE\_FLAG\_OVERLAPPED**, *lpOverlapped* is ignored.  If *hDevice* is opened with the **FILE\_FLAG\_OVERLAPPED** flag, the operation is performed as an overlapped (asynchronous) operation. In this case,*lpOverlapped* must point to a valid [**OVERLAPPED**](https://msdn.microsoft.com/en-us/library/ms684342(v=vs.85).aspx) structure that contains a handle to an event object. Otherwise, the function fails in unpredictable ways.  For overlapped operations, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns immediately, and the event object is signaled when the operation is complete. Otherwise, the function does not return until the operation is complete or an error occurs. |
|  |  |

**DELETE\_USN\_JOURNAL\_DATA structure**描述使用FSCTL\_DELETE\_USN\_JOURNAL控制代码删除更新序列号(USN)变更日志的信息。

|  |  |
| --- | --- |
| C++ | |
| typedef struct {  DWORDLONG UsnJournalID;  DWORD DeleteFlags;  } DELETE\_USN\_JOURNAL\_DATA, \*PDELETE\_USN\_JOURNAL\_DATA; | |
| UsnJournalID | 要删除的变更日志的标识符。如果日志是活动的，并且通过在DeleteFlags成员中设置USN\_DELETE\_FLAG\_DELETE标志来请求删除，那么这个标识符必须为当前卷指定变更日志。使用FSCTL\_QUERY\_USN\_JOURNAL检索此变更日志的标识符。如果标识符不是当前卷的变更日志，则FSCTL\_DELETE\_USN\_JOURNAL失败。如果仅通过在DeleteFlags中设置USN\_DELETE\_FLAG\_NOTIFY标志来请求通知而不是删除，则会忽略UsnJournalID。  The identifier of the change journal to be deleted.If the journal is active and deletion is requested by setting the USN\_DELETE\_FLAG\_DELETE flag in the DeleteFlags member, then this identifier must specify the change journal for the current volume. Use [FSCTL\_QUERY\_USN\_JOURNAL](https://msdn.microsoft.com/9491b054-934a-4b76-bf77-f397b6386f82) to retrieve the identifier of this change journal. If in this case the identifier is not for the current volume's change journal, [FSCTL\_DELETE\_USN\_JOURNAL](https://msdn.microsoft.com/6c85464d-019b-4923-9acf-152b4ee8c31b) fails.  If notification instead of deletion is requested by setting only the USN\_DELETE\_FLAG\_NOTIFY flag in DeleteFlags, UsnJournalID is ignored. |
| DeleteFlags | 指示是否执行删除或有关删除的通知，或两者都执行。DeleteFlags成员必须包含以下一个或两个值。  Indicates whether deletion or notification regarding deletion is performed, or both. The **DeleteFlags** member must contain one or both of the following values. |
|  |  |

参考：<https://docs.microsoft.com/zh-cn/windows/desktop/api/winioctl/ns-winioctl-delete_usn_journal_data>

**DeleteFlags**

|  |  |
| --- | --- |
| Value | Meaning |
| **USN\_DELETE\_FLAG\_DELETE**  0x00000001 | 如果设置了此标志而未设置USN\_DELETE\_FLAG\_NOTIFY标志，则[FSCTL\_DELETE\_USN\_JOURNAL](https://msdn.microsoft.com/6c85464d-019b-4923-9acf-152b4ee8c31b)操作将启动日志删除过程并立即返回。如果需要，日志删除过程将跨系统重新启动继续进行。  如果设置了此标志，并且还设置了USN\_DELETE\_FLAG\_NOTIFY标志，则会同时执行删除和通知操作。如果设置了此标志并且日志处于活动状态，则必须在UsnJournalID中为当前卷的变更日志提供标识符，否则操作将失败。如果日志不是活动的，则忽略UsnJournalID并删除日志。  If this flag is set and the USN\_DELETE\_FLAG\_NOTIFY flag is not set,  the[FSCTL\_DELETE\_USN\_JOURNAL](https://msdn.microsoft.com/6c85464d-019b-4923-9acf-152b4ee8c31b) operation starts the journal deletion process and returns immediately. The journal deletion process continues, if necessary, across system restarts.  If this flag is set and the USN\_DELETE\_FLAG\_NOTIFY flag is also set, both deletion and notification occur. If this flag is set and the journal is active, you must provide the identifier for the change journal for the current volume in UsnJournalID or the operation fails. If the journal is not active, then UsnJournalID is ignored and the journal is deleted. |
| **USN\_DELETE\_FLAG\_NOTIFY**  0x00000002 | 如果设置了此标志，则调用将设置关于何时完成删除的通知。当日志删除过程完成时，日志删除请求就完成了。如果设置了这个标志，而没有设置USN\_DELETE\_FLAG\_DELETE标志，那么调用将设置删除通知，该通知可能已经在进行中。例如，当应用程序启动时，它可能使用此标志来确定是否正在进行删除。  如果设置了这个标志，同时也设置了USN\_DELETE\_FLAG\_DELETE标志，那么删除和通知都会发生。通知使用I/O完成端口或异步事件通知的其他机制执行。  If this flag is set, the call sets up notification about when deletion is complete. The journal deletion request is completed when the journal deletion process is complete. If this flag is set and the USN\_DELETE\_FLAG\_DELETE flag is not set, then the call sets up notification of a deletion that may already be in progress. For example, when your application starts, it might use this flag to determine if a deletion is in progress.  If this flag is set and the USN\_DELETE\_FLAG\_DELETE flag is also set, both deletion and notification occur. The notification is performed using an I/O completion port or another mechanism for asynchronous event notification. |
|  |  |

**Return code**

|  |  |
| --- | --- |
| Value | Meaning |
| **ERROR\_INVALID\_FUNCTION** | 指定的卷不支持变更日志。  The specified volume does not support change journals. |
| **ERROR\_INVALID\_PARAMETER** | 一个或多个参数无效，例如，如果提供的句柄不是卷句柄，DeviceIoControl将返回此错误代码。  One or more parameters is invalid, for example, [**DeviceIoControl**](https://msdn.microsoft.com/en-us/library/aa363216(v=vs.85).aspx) returns this error code if the handle supplied is not a volume handle. |
| **ERROR\_JOURNAL\_DELETE\_IN\_PROGRESS** | 尝试在日志删除过程中读取、创建、删除或修改日志，或者尝试在日志删除过程中写入USN记录。  An attempt is made to read from, create, delete, or modify the journal while a journal deletion is in process, or an attempt is made to write a USN record while a journal deletion is in process. |
|  |  |

|  |
| --- |
| 备注 |
| 如果操作成功完成，DeviceIoControl返回一个非零值。  如果操作失败或挂起，DeviceIoControl返回零。要获取错误信息，请调用GetLastError。可能的返回值参见上表。  可以使用FSCTL\_DELETE\_USN\_JOURNAL删除变更日志。NTFS文件系统启动删除操作并立即返回到调用进程，除非在DELETE\_USN\_JOURNAL\_DATA的DeleteFlags成员中设置了USN\_DELETE\_FLAG\_NOTIFY标志。  如果同时设置了USN\_DELETE\_FLAG\_NOTIFY和USN\_DELETE\_FLAG\_DELETE标志，  则调用FSCTL\_DELETE\_USN\_JOURNAL开始删除过程。然后调用阻塞调用线程并等待删除(在同步或非重叠调用上)，或者使用I/O完成端口或其他机制设置事件通知，并返回(在异步或重叠调用上)。  您还可以使用FSCTL\_DELETE\_USN\_JOURNAL来接收变更日志删除完成的通知，方法是只设置**USN\_DELETE\_FLAG\_NOTIFY**。如果这样做，FSCTL\_DELETE\_USN\_JOURNAL操作要么等到删除完成后才返回(在同步或非重叠调用中)，要么通过使用I/O完成端口或其他机制(在异步或重叠调用中)设置事件通知。  应用程序接收通知的删除可能是由当前进程或其他进程发起的。例如，当应用程序启动时，它可以使用FSCTL\_DELETE\_USN\_JOURNAL来确定由其他进程启动的删除是否正在进行中，如果正在进行，则退出。  完全删除变更日志需要扫描变更日志所在的卷，对于包含许多文件的卷，这可能需要很长时间。即使跨系统重新启动，操作也会继续完成。尝试在删除过程中创建、修改、删除或查询变更日志时失败，并返回错误代码ERROR\_JOURNAL\_DELETE\_IN\_PROGRESS。  FSCTL\_DELETE\_USN\_JOURNAL操作有显著的性能开销，因此应该谨慎使用。当当前USN值接近可能的最大USN值时，管理员应该删除日志。  You can use **FSCTL\_DELETE\_USN\_JOURNAL** to delete a change journal. The NTFS file system starts a deletion operation and returns immediately to the calling process, unless the **USN\_DELETE\_FLAG\_NOTIFY** flag is set in the **DeleteFlags** member of [**DELETE\_USN\_JOURNAL\_DATA**](https://msdn.microsoft.com/en-us/library/aa363928(v=vs.85).aspx).  If the **USN\_DELETE\_FLAG\_NOTIFY** and **USN\_DELETE\_FLAG\_DELETE** flags are both set, a call to **FSCTL\_DELETE\_USN\_JOURNAL** begins the deletion process. Then the call either blocks the calling thread and waits for the deletion (on a synchronous or non-overlapped call), or sets up event notification by using an I/O completion port or other mechanism, and returns (on an asynchronous or overlapped call).  You can also use **FSCTL\_DELETE\_USN\_JOURNAL** to receive notification that a change journal deletion is complete, by setting only**USN\_DELETE\_FLAG\_NOTIFY**. If you do so, the **FSCTL\_DELETE\_USN\_JOURNAL** operation either waits until the deletion completes before returning (on a synchronous or non-overlapped call), or sets up event notification by using an I/O completion port or other mechanism (on an asynchronous or overlapped call).  The deletion on which an application receives notification may have been initiated by the current process, or some other process. For example, when an application is started, it can use **FSCTL\_DELETE\_USN\_JOURNAL** to determine if a deletion started by some other process is in progress and if it is, exit.  Complete deletion of a change journal requires a scan of the volume where the change journal resides, which may take a long time on a volume with many files. The operation continues to completion even across system restarts. Attempts to create, modify, delete, or query the change journal while deletion is in progress fail and return the error code **ERROR\_JOURNAL\_DELETE\_IN\_PROGRESS**.  The **FSCTL\_DELETE\_USN\_JOURNAL** operation has a significant performance cost, so it should be used sparingly. An administrator should delete a journal when the current USN value approaches that of the maximum possible USN value. |

|  |
| --- |
| 示例代码 |
| #include <windows.h>  #include <tchar.h>  DELETE\_USN\_JOURNAL\_DATA dujd;  dujd.UsnJournalID = UsnInfo.UsnJournalID;  dujd.DeleteFlags = USN\_DELETE\_FLAG\_DELETE;  status = DeviceIoControl(hVol,  FSCTL\_DELETE\_USN\_JOURNAL,  &dujd,  sizeof(dujd),  NULL,  0,  &br,  NULL);  if (0 != status){  printf("成功删除USN日志文件!\n");  }  else{  printf("删除USN日志文件失败 error:%d\n", GetLastError());  } |

七、关闭卷标

当日志读取完成以后，必须关闭卷，释放资源。API如下：

由于只有NTFS文件系统才支持$UsnJrnl日志，因此在读取日志前需要判断卷标的格式。API如下：

|  |
| --- |
| C++ |
| BOOL CloseHandle(  HANDLE hObject  ); |

参考：<https://docs.microsoft.com/en-us/windows/desktop/api/handleapi/nf-handleapi-closehandle>

|  |  |  |
| --- | --- | --- |
| hObject | | 打开对象的有效句柄。 |
|  | |  |
| 备注 | | |
| 如果函数成功，返回值为非零。  如果函数失败，返回值为零。要获取错误信息，请调用GetLastError。  CloseHandle函数可以关闭下列对象：  访问令牌（Access token）  通讯设备（Communications device）  控制台输入（Console input）  控制台屏幕缓冲区（Console screen buffer）  事件（Event）  文件（File）  文件映射（File mapping）  I / O完成端口（I/O completion port）  工作（Job）  邮槽（Mailslot）  内存资源的通知（Memory resource notification）  互斥锁（Mutex）  命名管道（Named pipe）  管道（Pipe）  进程（Process）  信号量（Semaphore）  线程（Thread）  事务（Transaction）  可等待定时（Waitable timer） | | |