9. Dateisysteme (6)

- 9.7 Standard-Dateisysteme: Ext2/Ext3 und NTFS

NTFS

- NTFS is the native file system format of Windows
- NTFS uses 64-bit cluster (block) indexes
  - Theoretical ability to address volumes of up to 16 exabytes (16 billion GByte)
  - Windows 2000 limits the size of an NTFS volume to that addressable with 32-bit clusters, which is 128 TByte (using 64-KByte clusters)
- Why use NTFS instead of FAT? FAT is simpler, making it faster for some operations, but NTFS supports:
  - Larger file sizes and disks
  - Better performance on large disks, large directories, and small files
  - Reliability, Security

9.7 Standarddateisysteme: Windows NTFS

NTFS Recoverability

PC disk I/O in the old days: Speed was most important
NTFS changes this view – Reliability counts most:

- I/O operations that alter NTFS structure are implemented as atomic transactions
  - Change directory structure,
  - extend files, allocate space for new files
- Transactions are either completed or rolled back
- NTFS uses redundant storage for vital FS information
  - Contrasts with FAT / HPFS on-disk structures, which have single sectors containing critical file system data
  - Read error in these sectors -> volume lost

NTFS Security and Recoverability

NTFS security is derived from Windows object model

- Open file is implemented as file object; security descriptor is stored on disk as part of the file
- NT security system verifies access rights when a process tries to open a handle to any object
- Administrator or file owner may set permissions

NTFS recoverability ensures integrity of FS structure

- No guarantees for complete recovery of user files
- Layered driver model + FTDISK (**f**ault **t**olerant) driver
  - Mirroring of data – RAID level 1
  - Striping of data – RAID level 5 (one disk with parity info)

Large Disks and Large Files

- NTFS enumerates clusters with 64-bit numbers
- Up to 2^{64} clusters of up to 64 KBytes size
- Maximum file size: 2^{64} bytes
- Cluster size is adjustable
  - 512 bytes on small disks
  - Maximum of 64 KByte on large disks
- Multiple data streams
  - File info: name, owner, time stamps, type implemented as attribute
  - Each attribute consists of a stream – sequence of bytes
  - Default data stream has no name
  - New streams can be added: *myfile.dat:stream2*
  - File operations manipulate all streams simultaneously

Other NTFS Features

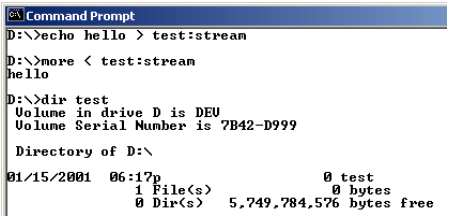
- Multiple data streams
- Unicode-based names
- Hard links
- Junctions
- Compression and sparse files
- Change logging
- Per-user volume quotas
- Link tracking
- Encryption
- POSIX support
- Defragmentation

Multiple Data Streams (1)

- In NTFS, each unit of information associated with a file, including its name, its owner, its time stamps, its contents, and so on, is implemented as a file attribute (NTFS object attribute)
- Each attribute consists of a single *stream,* that is, a simple sequence of bytes
  - This generic implementation makes it easy to add more attributes (and therefore more streams) to a file
  - Since a file's data is “just another attribute” of the file and because new attributes can be added, NTFS files (and file directories) can contain multiple data streams

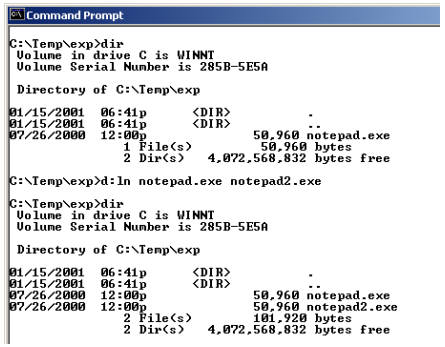
Multiple Data Streams (2)

- An NTFS file has one default data stream, which has no name
  - An application can create additional, named data streams and access them by referring to their names.
  - To avoid altering the Microsoft Windows I/O APIs, which take a string as a filename argument, the name of the data stream is specified by appending a colon (:) to the filename, e.g. *myfile:stream2*



Hard Links

- A hard link allows multiple paths to refer to the same file or directory
  - If you create a hard link named *C:\Users\Documents\Spec.doc* that refers to the existing file *C:\My Documents\Spec.doc*, the two paths link to the same on-disk file and you can make changes to the file using either path
  - can create hard links with the Windows API *CreateHardLink* function or the *ln* POSIX function

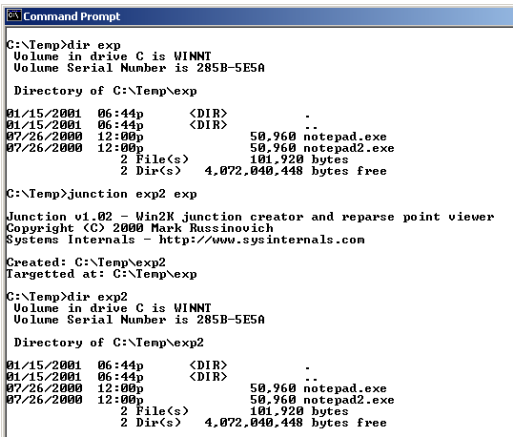


Junctions (1)

- Junctions, also called symbolic links, allow a directory to redirect file or directory pathname translation to an alternate directory
  - If the path *C:\Drivers* is a junction that redirects to *C:\Winnt\System32\Drivers*, an application reading *C:\Drivers\Ntfs.sys* actually reads *C:\Winnt\System\Drivers\Ntfs.sys*
  - Junctions are a useful way to lift directories that are deep in a directory tree to a more convenient depth without disturbing the original tree's structure or contents

Junctions (2)

- You can create junctions with the *junction* tool from Sysinternals or the *linkd* tool from the Resource Kits



Link Tracking (1)

- Several types of symbolic file links are used by layered applications
  - Shell shortcuts allow users to place files in their shell namespace (on their desktop, for example) that link to files located in the file system namespace
  - Object linking and embedding (OLE) links allow documents from one application to be transparently embedded in the documents of other applications
- In the past, these links were difficult to manage
  - If someone moved a link source (what a link points to), the link broke

Link Tracking (2)

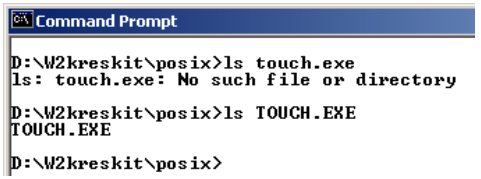
- Windows now has a link-tracking service: **TrkWks** (which runs in *services.exe*) tags link sources with a unique object ID
  - NTFS can return the name of a file, given a link; so if the link moves, the service can query each of a system's volumes for the object ID
  - A distributed link-tracking service, **TrkSvr**, works to track link source movement across systems

Encryption

- While NTFS implements security for files and directories, the security is ineffective if the physical security of the computer is compromised
  - Can install a parallel copy of Windows
  - NTFSDOS, Linux NTFS driver
- Encrypting File System (EFS)
  - Like compression, its operation is transparent
  - Also like compression, encryption is a file and directory attribute
  - Files that are encrypted can be accessed only by using the private key of an account's EFS private/public key pair, and private keys are locked using an account's password

POSIX Support

- POSIX support requires two file system features:
  - Primary group in security descriptor
  - Case-sensitive names



Compression and Sparse Files

- NTFS supports transparent compression of files
  - When a directory is marked compressed, it means that any files or subdirectories are marked compressed
  - Compression is performed on 16-cluster blocks of a file
  - Use Explorer or the *compact* tool to compress files (*compact* shows compression ratios for compressed files)
- Sparse files are an application-controlled form of compression. They define parts of a file as empty – those areas don't occupy any disk space
  - Applications use Windows API to define empty areas

NTFS On-Disk Structure

- Volumes correspond to logical partitions on disk
- Fault tolerant volumes may span multiple disks
  - Windows 2000 Disk Administrator utility
- Volume consists of series of files + unallocated space
  - FAT volume: some areas specially formatted for file system
  - NTFS volume: all data are stored as ordinary files
- NTFS refers internally to clusters
  - Cluster factor: #sectors/cluster; varies with volume size; (integral number of physical sectors; always a power of 2)
- Logical Cluster Numbers (LCNs):
  - refer to physical location
  - LCNs are contiguous enumeration of all clusters on a volume

NTFS Cluster Size

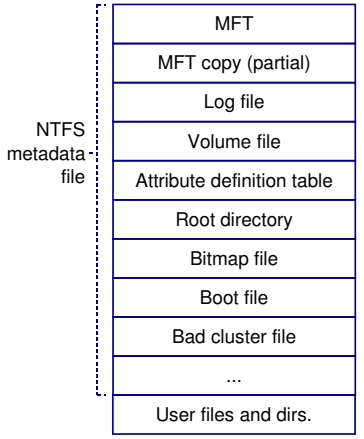
- Default cluster size is disk-size dependent
  - 512 bytes for small disks (up to 512 MByte)
  - 1 KByte for disks up to 1 GByte
  - 2 KByte for disks between 1 and 2 GByte
  - 4 KByte for disks larger than 2 GByte
- Tradeoff: disk fragmentation vs. wasted space
- NTFS refers to physical locations via LCNs
  - Physical cluster = LCN * cluster-factor
- Virtual Cluster Numbers (VCNs):
  - Enumerates clusters belonging to a file; mapped to LCNs
  - VCNs are not necessarily physically contiguous

Master File Table

All data stored on a volume is contained in a file

MFT: Heart of NTFS volume structure

- Implemented as array of file records
- One row for each file on the volume (including one row for MFT itself)
- Metadata files store file system structure information (hidden files: *$MFT*; *$Volume*...)
- More than one MFT record for highly fragmented files



NTFS operation

Mounting a volume

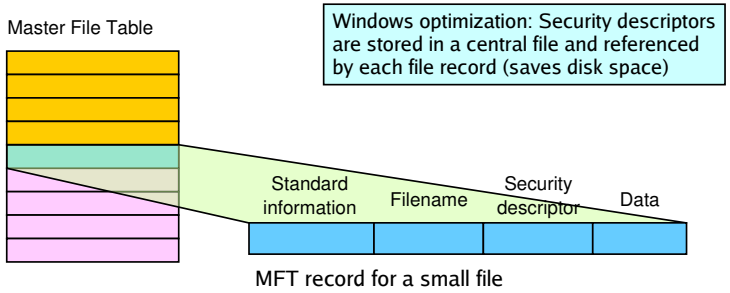
- NTFS looks in boot file for physical address of MFT (*$MFT*)
- 2nd entry in MFT points to copy of MFT (*$MFTMirr*)
  - used to locate metadata files if MFT is corrupted
- MFT entry in MFT contains VCN-to-LCN mapping info
- NTFS obtains from MFT addresses of metadata files
  - NTFS opens these files
- NTFS performs recovery operations
- File system is now ready for user access

File Records & File Reference Numbers

- File on NTFS volume is identified by **file reference**
  - File number == index in MFT
  - Sequence number – used by NTFS for consistency checking; incremented each time a reference is re-used
- File Records:
  - File is collection of attribute/value pairs (one of which is data)
  - Unnamed data attribute (standard stream)
  - Other attributes: filename, time stamp, security descriptor, ...
  - Each file attribute is stored as separate stream of bytes within a file

File Records (contd.)

- NTFS doesn't read/write files:
  - It reads/writes attribute streams
  - Operations: create, delete, read (byte range), write (byte range)
  - Read/write normally operate on unnamed data attribute



Standard Attributes for NTFS Files

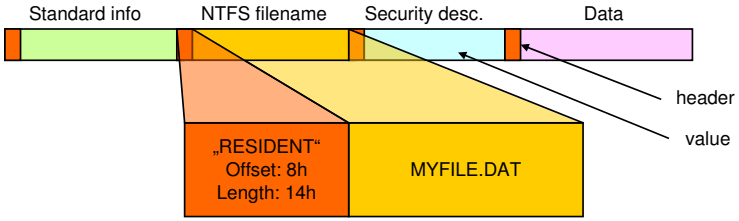
Attribute	Description
Standard information	File attributes: read-only, archive, etc; time stamps; creation/modification time; hard link count
Filename	Name in Unicode characters; multiple filename attributes possible (POSIX links); short names for access by MS-DOS and 16-bin Win applications
Security descriptor	Specifies who owns the file and who can access it
data	Contents of the file; a file has one default unnamed data attribute; directory has no default data attrib.
Index root, index	Three attributes used to implement filename allocation, bitmap index for large directories (dirs. only)
Attribute list	List of attributes that make up the file and first reference of the MFT record in which the attribute is located (for files which require multiple MFT file records)

Attributes (contd.)

- Each attribute in a file record has a name and a value
- NTFS identifies attributes:
  - Uppercase name starting with $: *$FILENAME*, *$DATA*
- Attribute's value: Byte stream
  - The filename for *$FILENAME*
  - The data bytes for *$DATA*
- Attribute names correspond to numeric typecodes
- File attributes in an MFT record are ordered by typecodes
  - Some attribute types may appear more than once (e.g. Filename)

Resident & Nonresident Attributes

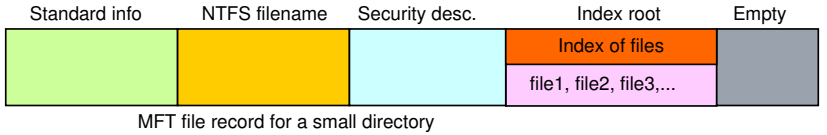
- Small files:
  - All attributes and values fit into MFT
  - Attribute with value in MFT is called „resident“
  - All attributes start with header (always resident)
  - Header contains offset to attribute value and length of value



Attributes (contd.)

Small directory:

- *index root* attribute contains index of file references for files and subdirectories



MFT file record for a small directory

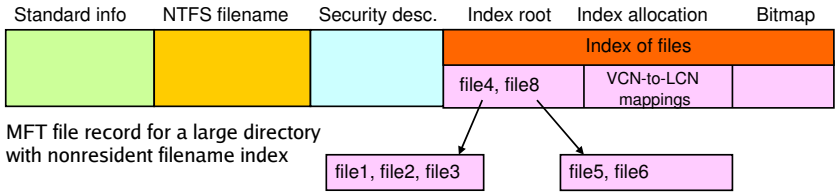
If file attribute does not fit into MFT:

- NTFS allocates separate cluster (run, extent) to store the values
- NTFS allocates additional runs if an attribute's value later grows
- Those attributes are called „non-resident“
- Header of non-resident attribute contains location info

Large files & directories (1)

MFT record for large file with 2 data runs

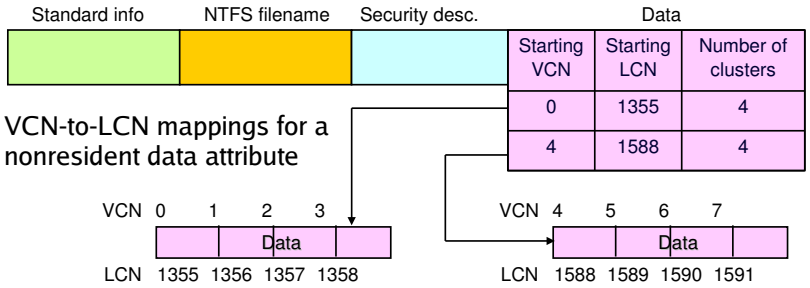
- Only attributes that can grow can be non-resident
- Filename & standard info are always resident
- Index of files for directories forms B+ tree



MFT file record for a large directory with nonresident filename index

Large files & directories (2)

- NTFS keeps track of runs by means of VCNs (Virtual Cluster Numbers)
  - Logical Cluster Numbers represent an entire volume
  - Virtual Cluster Numbers represent clusters belonging to one file
  - Attribute lists may extend over multiple runs (not only data)

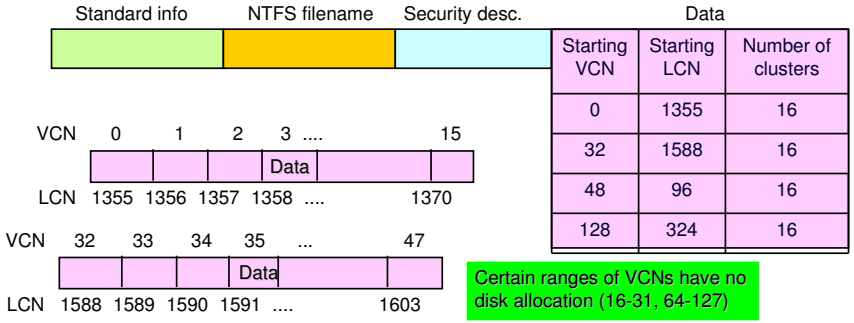


Data Compression

- NTFS supports compression
  - Per-file, per-directory, per-volume basis
  - NTFS compression is performed on user data only, not NTFS metadata
  - transparent
- Inspect files/volume via Windows API:
  `GetVolumeInformation()`,
  `GetCompressedFileSize()`
- Change settings for files/directories:
  `DeviceIoControl()`
  with flags
  `FSCTL_GET_COMPRESSION, FSCTL_SET_COMPRESSION`

Compression of sparse files

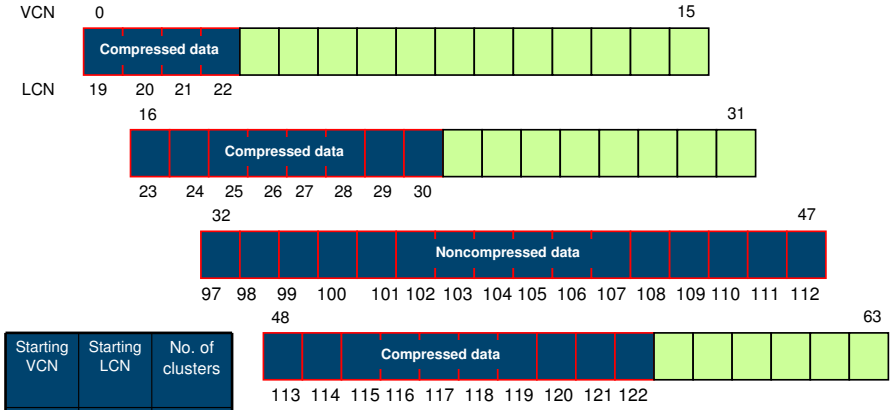
- NTFS zeroes all file contents on creation
- Many sparse files contain large amount of zero-bytes
  - These bytes occupy space on disk – unless files are compressed



Compressing Nonsparse Data

- NTFS divides the file's unprocessed data into *compression units* 16 clusters long
- Certain sequence might not compress much
  - NTFS determines for each compression unit whether it will shrink by at least one cluster
  - If data does not compress, NTFS allocates cluster space and simply writes data
  - If data compresses by at least one cluster, NTFS allocates only the clusters needed for compressed data
- When writing data, NTFS ensures that each run begins on virtual 16-cluster boundary
  - NTFS reads/writes at least one compression unit when accessing a file
  - Read-ahead + asynch. decompression improve performance

Data runs of a compressed file



Starting VCN	Starting LCN	No. of clusters
0	19	4
16	23	8
32	97	16
48	113	10

MFT record for a compressed file

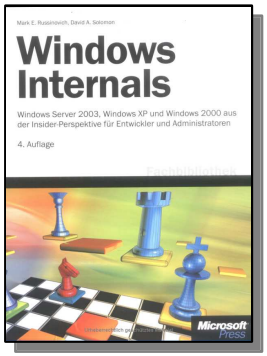
Literatur

Mark E. Russinovich, David A. Solomon,

Microsoft Windows Internals. Windows 2000, Windows XP und Windows Server 2003,

4. Auflage, Microsoft Press, 2005

- File Systems supported by Windows
- File System Driver Architecture
- NTFS Design Goals and Features
- NTFS On-Disk Structure



Alles hat ein Ende

... auch „Betriebssysteme II“ ;-)

Nächsten Dienstag: Zusammenfassung BS II
Montag / Mittwoch: Zusammenfassung BS I