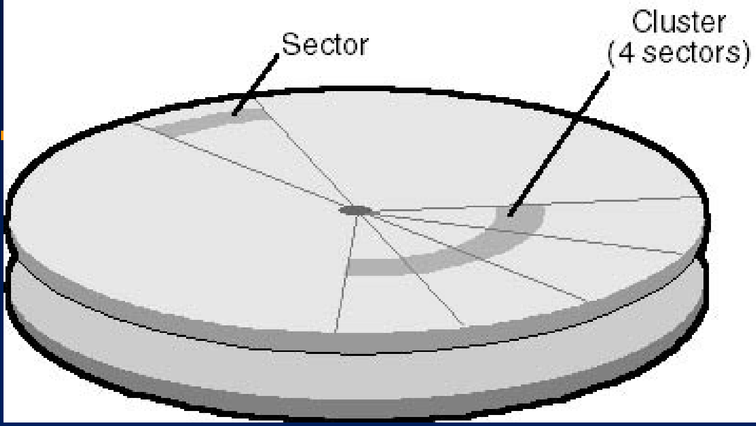
Unit 8: File System

8.2. Windows File Systems

Roadmap for Section 8.2

- File Systems supported by Windows
- NTFS Design Goals
- File System Driver Architecture
- NTFS Operation
- Windows File System On-Disk Structure
- NTFS File Compression

Windows File System Terminology



- Sectors:
  - hardware-addressable blocks on a storage medium
  - Typical sector size on hard disks for x86-based systems was 512 bytes, now is 4KB
- File system formats:
  - Define the way data is stored on storage media
  - Impact a file system features: permissions & security, limitations on file size, support for small/large files/disks
- Clusters:
  - Addressable blocks that many file system formats use
  - Cluster size is always a multiple of the sector size
  - Cluster size tradeoff: space efficiency vs. access speed
- Metadata:
  - Data stored on a volume in support of file system format management
  - Metadata includes the data that defines the placement of files and directories on a volume, for example
  - Typically not accessible to applications

Formats Supported by Windows

- CD-ROM File System (CDFS)
- Universal Disk Format (UDF)
- File Allocation Table (FAT12, FAT16, FAT32, and exFAT)
- New Technology File System (NTFS)
- Resilient File System (ReFS), introduced first in Windows Server 2012 and then in Windows 8.1

CDFS

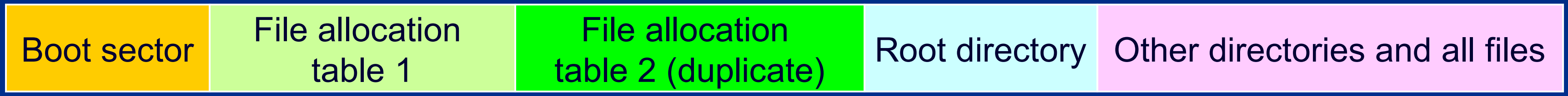
- CDFS is a relatively simple format that was defined in 1988 as the read-only formatting standard for CD-ROM media (ISO 9660).
- Windows 2000 implemented ISO 9660-compliant CDFS in \Winnt\System32\Drivers\Cdfs.sys, with long filename support defined by Level 2 of the ISO 9660 standard
- Because of its simplicity, the CDFS format has a number of restrictions
  - Directory and file names must be fewer than 32 characters long
  - Directory trees can be no more than eight levels deep
- CDFS is considered a legacy format because the industry has adopted the Universal Disk Format (UDF) as the standard for read-only media

UDF

- OSTA (Optical Storage Technology Association) defined UDF in 1995 as a format to replace CDFS for magneto-optical storage media, mainly DVD-ROM (ISO 13346)
  - The Windows 2000 UDF file system implementation is ISO 13346-compliant and supports UDF versions 1.02 and 1.5
- UDF file systems have the following traits:
  - Filenames can be 255 characters long
  - The maximum path length is 1023 characters
- Although the UDF format was designed with rewritable media in mind, in Windows 2000 the UDF driver (\Winnt\System32\Drivers\Udfs.sys) provided read-only support

FAT

- FAT (File Allocation Table) file systems are a legacy format that originated in DOS and Windows 9x
- Reasons why Windows supports FAT file systems:
  - to enable upgrades from other versions of Windows
  - compatibility with other operating systems in multiboot systems
  - as an universal format for floppy disks, USB sticks & flash memory cards
- Windows FAT file system driver is implemented in \Winnt\System32\Drivers\Fastfat.sys
- Each FAT format includes a number that indicates the number of bits the format uses to identify clusters on a disk


Boot sector
File allocation table 1
File allocation table 2 (duplicate)
Root directory
Other directories and all files

FAT format organization

FAT12

- FAT12's 12-bit cluster identifier limits a partition to storing a maximum of 2^{12} (4096) clusters
  - Windows uses cluster sizes from 512 bytes to 8 KB in size, which limits a FAT12 volume size to 32 MB
  - Windows uses FAT12 as the format for all 5-inch floppy disks and 3.5-inch floppy disks, which store up to 1.44 MB of data

FAT16

- FAT16, with a 16-bit cluster identifier, can address 2^{16} (65,536) clusters
  - On Windows, FAT16 cluster sizes range from 512 bytes (the sector size) to 64 KB, which limits FAT16 volume sizes to 4 GB
  - The cluster size Windows uses depends on the size of a volume

FAT32

- FAT32 uses 32-bit cluster identifiers, but reserves the high 4 bits, so in effect it has 28-bit cluster identifiers
  - It was included with Windows 95 OSR2, Windows 98, and Windows Millennium Edition
  - Because FAT32 cluster sizes can be as large as 32 KB, FAT32 has a theoretical ability to address 8 TB volumes
  - Although Windows works with existing FAT32 volumes of larger sizes (created in other operating systems), it limits new FAT32 volumes to a maximum of 32 GB
  - FAT32's higher potential cluster numbers let it more efficiently manage disks than FAT16; it can handle up to 128-MB volumes with 512-byte clusters
- Unlike FAT12 and FAT16, root directory is not fixed size or location in FAT32
  - Largest file size on Windows NT is 4GB (and on Windows 9x is 2GB)

exFAT

- exFAT is the most recently defined FAT-based file system format
  - Was introduced in 2006, been optimized for flash memory such as USB flash drives and SD cards. (It is proprietary and Microsoft owns patents on several elements of its design.)
  - exFAT allows individual files larger than 4 GiB (file size limit of 16 EiB − 1 byte)
  - The cluster size is up to 32 MiB
  - exFAT introduces metadata integrity through the use of checksums
  - exFAT does not support short (8.3 format) filenames

NTFS

- NTFS is the native file system format of Windows (*Note*: Windows NT, do not confuse with Windows 9x !)
- NTFS uses 64-bit cluster indexes
  - Theoretical ability to address volumes of up to 16 exabytes (16 billion GB)
  - Windows 2000 limits the size of an NTFS volume to that addressable with 32-bit clusters, which is 128 TB (using 64-KB 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

CIFS – the Common Internet File System

- The standard Windows network file system
- The file sharing protocol at the heart of CIFS is an updated version of the Server Message Block (SMB) protocol
  - dates back to the mid-1980s
  - in 1996/97, Microsoft submitted draft CIFS specifications to the IETF
- The SMB protocol was originally developed to run over NetBIOS (Network Basic Input Output System) LANs
  - Until Windows 2000, NetBIOS support was required for SMB transport
  - The machine and service names visible in the Windows Network Neighborhood are, basically, NetBIOS addresses (Windows 2000 and later use DNS names)
- Windows 3.11 (WfW) introduced:
  - service announcement and location system called Browsing
  - The browser service provides the list of available file and print services presented in the Network Neighborhood
  - Workgroup concept was expanded to create NT Domains

File System Format Compatibility

- FAT12/FAT16 supported on all Microsoft OS's
- FAT32:
  - Supported only from Windows 2000
  - Winternals FAT32 driver for NT4.0
- NTFS:
  - Supported only on Windows NT-based OS's
  - Winternals NTFSDOS driver for DOS
  - Winternals NTFS for Windows 98 driver for Win9x/Me

NTFS Design Goals

- Overcome limitations inherent in FAT / HPFS
  - FAT (File Allocation Table) does not support large disks very well
  - FAT16 (MS-DOS file system) supports only up to 2^{16} clusters and 2 GB disks (with 64 Kb clusters!!)
  - FAT / root directory represents single point of failure
  - Number of entries in root directory is limited
  - OS/2 HPFS (High Performance File System, developed by Microsoft & IBM in 1989) removed some of FAT's limitations, but still did not support recoverability, security, data redundancy, and fault-tolerance
    (Later versions of HPFS supported up to 2TB disks.)

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, e.g.:
    - 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 is 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 driver
  - Mirroring of data – RAID level 1
  - Striping of data – RAID level 5 (one disk with parity info)

Large Disks and Large Files

- Efficient support for large files and disks in NTFS
- FAT16:
  - 16-bit wide table stores allocation status of disk
  - Up to 65.536 clusters per volume (#files !!); adjustable cluster size
- FAT32:
  - New introduced since Windows 2000
  - 4KB clusters on volumes up to 8 GB
  - Can relocate root directory / use backup copy of FAT
  - Root directory is ordinary cluster chain – no limits on #entries
- HPFS (support dropped in NT 4.0):
  - 32 bits to enumerate allocation units; maximum file size: 4GB
  - Allocates disk space in terms of physical sectors of 512 bytes; problem with some disks (1024 bit sectors)

Large Disks and Large Files (contd.)

- 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 64Kb on large disks
- Multiple data streams
  - File info: filename, owner, time stamps, type, etc. are implemented as attributes
  - 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

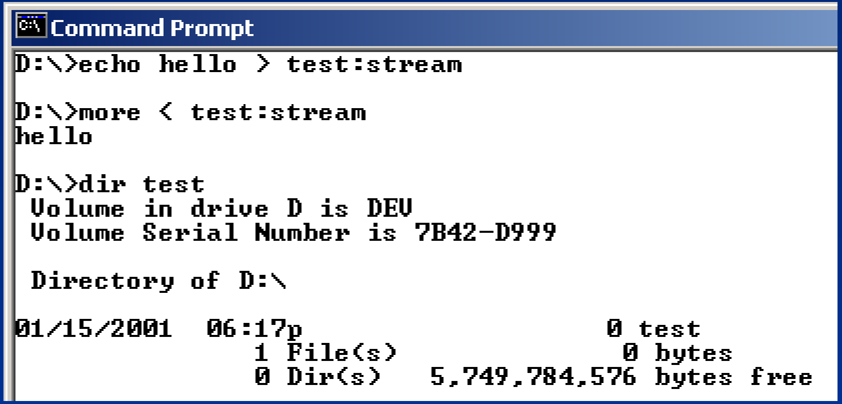
- Multiple data streams
- Unicode-based names
- Hard links
- Junctions
- Compression and sparse files
- Change logging
- Per-user volume quotas
- Link tracking
- Encryption
- POSIX support
- Defragmentation

Multiple Data Streams

- 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
    - Because 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

- 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


Command Prompt

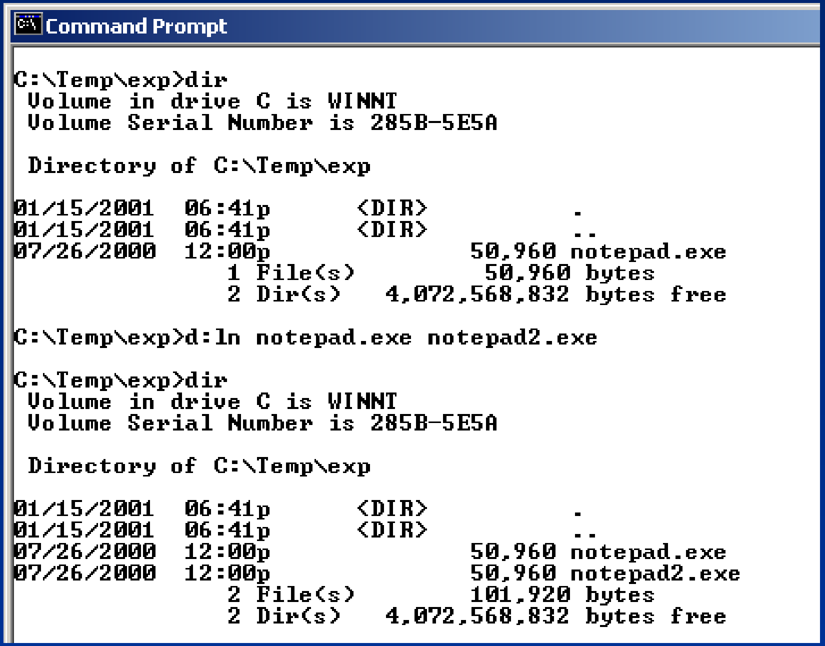
```
D:\>echo hello > test:stream

D:\>more < test:stream
hello

D:\>dir test
 Volume in drive D is DEV
 Volume Serial Number is 7B42-D999

 Directory of D:\

01/15/2001  06:17p                   0 test
               1 File(s)              0 bytes
               0 Dir(s)   5,749,784,576 bytes free
```

Unicode Names

- Like Windows as a whole, NTFS is fully Unicode enabled, using Unicode characters to store names of files, directories, and volumes

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



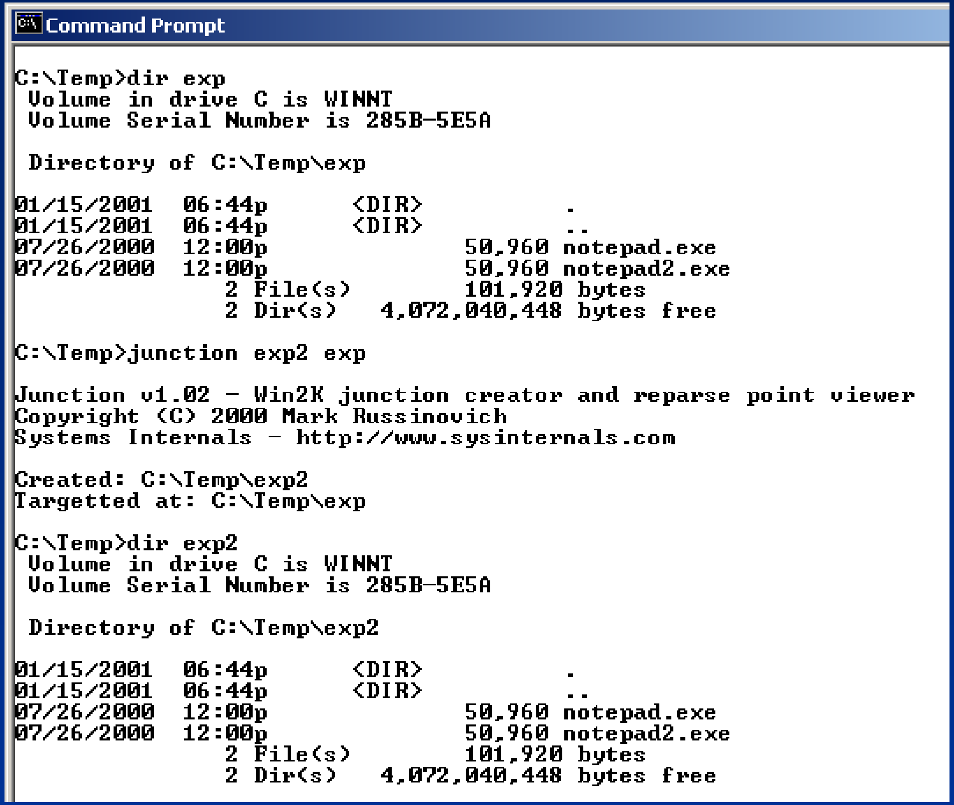
```
C:\Temp\exp>dir
 Volume in drive C is WINNT
 Volume Serial Number is 285B-5E5A

 Directory of C:\Temp\exp

01/15/2001  06:41p      <DIR>          .
01/15/2001  06:41p      <DIR>          ..
07/26/2000  12:00p              50,960 notepad.exe
               1 File(s)         50,960 bytes
               2 Dir(s)   4,072,568,832 bytes free

C:\Temp\exp>d:ln notepad.exe notepad2.exe

C:\Temp\exp>dir
 Volume in drive C is WINNT
 Volume Serial Number is 285B-5E5A

 Directory of C:\Temp\exp

01/15/2001  06:41p      <DIR>          .
01/15/2001  06:41p      <DIR>          ..
07/26/2000  12:00p              50,960 notepad.exe
07/26/2000  12:00p              50,960 notepad2.exe
               2 File(s)        101,920 bytes
               2 Dir(s)   4,072,568,832 bytes free
```

Junctions

- 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

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



Command Prompt

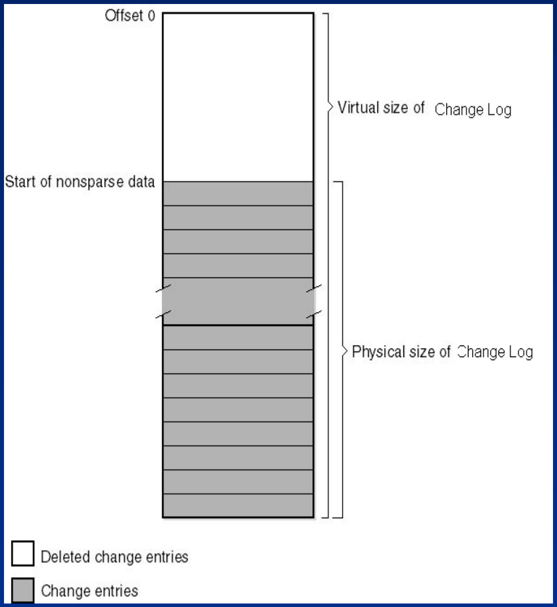
```
C:\Temp>dir exp
 Volume in drive C is WINNT
 Volume Serial Number is 285B-5E5A

 Directory of C:\Temp\exp

01/15/2001  06:44p      <DIR>          .
01/15/2001  06:44p      <DIR>          ..
07/26/2000  12:00p              50,960 notepad.exe
07/26/2000  12:00p              50,960 notepad2.exe
               2 File(s)        101,920 bytes
               2 Dir(s)   4,072,040,448 bytes free

C:\Temp>junction exp2 exp

Junction v1.02 - Win2K junction creator and reparse point viewer
Copyright (C) 2000 Mark Russinovich
Systems Internals - http://www.sysinternals.com

Created: C:\Temp\exp2
Targetted at: C:\Temp\exp

C:\Temp>dir exp2
 Volume in drive C is WINNT
 Volume Serial Number is 285B-5E5A

 Directory of C:\Temp\exp2

01/15/2001  06:44p      <DIR>          .
01/15/2001  06:44p      <DIR>          ..
07/26/2000  12:00p              50,960 notepad.exe
07/26/2000  12:00p              50,960 notepad2.exe
               2 File(s)        101,920 bytes
               2 Dir(s)   4,072,040,448 bytes free
```

Change Logging

- Many types of applications, such as incremental backup utilities, need to monitor a volume for changes
- An obvious way to watch for changes is to perform a full scan
  - Very performance inefficient
- There is a way for an application to “wait” on a directory and be told of notifications
  - An application can miss changes since it must specify a buffer to hold them

Change Logging

- In Windows 2000, NTFS introduced the change log file, which is a sparse metadata file that records file system events (not enabled by default)
  - As the file exceeds its maximum on-disk size, NTFS frees the disk space for the oldest portions marking them empty
  - An application uses Win32 APIs to read events
  - The log file is shared, and generally large enough that an application won't miss changes even during heavy file system activity



Per-User Volume Quotas

- NTFS quota-management support allows for per-user specification of quota enforcement
  - Can be configured to log an event indicating the occurrence to the system Event Log if a user surpasses his warning limit
  - If a user attempts to use more volume storage then her quota limit permits, NTFS can log an event to the system Event Log and fail the application file I/O that would have caused the quota violation with a "disk full" error code
- User disk space is tracked on a per-volume basis by summing the *logical* sizes of all the files and directories that have the user as the owner in their security descriptors

Link Tracking

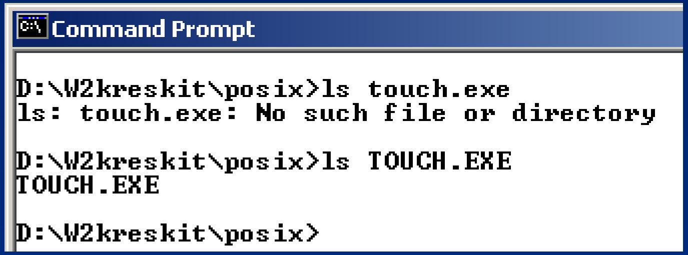
- Several types of symbolic file links are used by layered applications
  - Shell shortcuts allow users to place files in their shell namespace (on their desktop, for example) that link to files located in the file system namespace
  - Object linking and embedding (OLE) links allow documents from one 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
- Windows now has a link-tracking service, TrkWks (it 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
  - Can boot with Linux live CDs and access NTFS partitions
- 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
- While you might think that its implemented as a file system filter driver, it's a driver that's tightly connected to NTFS driver

POSIX Support

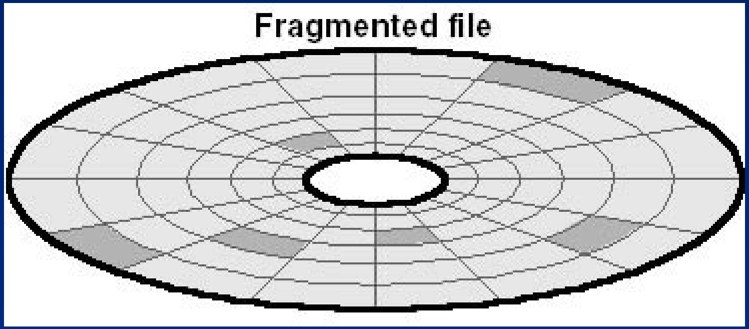
- POSIX support requires two file system features:
  - Primary group in security descriptor
  - Case-sensitive names

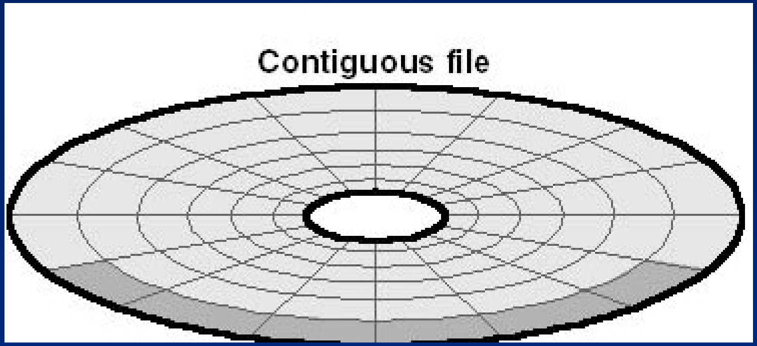


```
D:\W2kreskit\posix>ls touch.exe
ls: touch.exe: No such file or directory

D:\W2kreskit\posix>ls TOUCH.EXE
TOUCH.EXE

D:\W2kreskit\posix>
```

Defragmentation

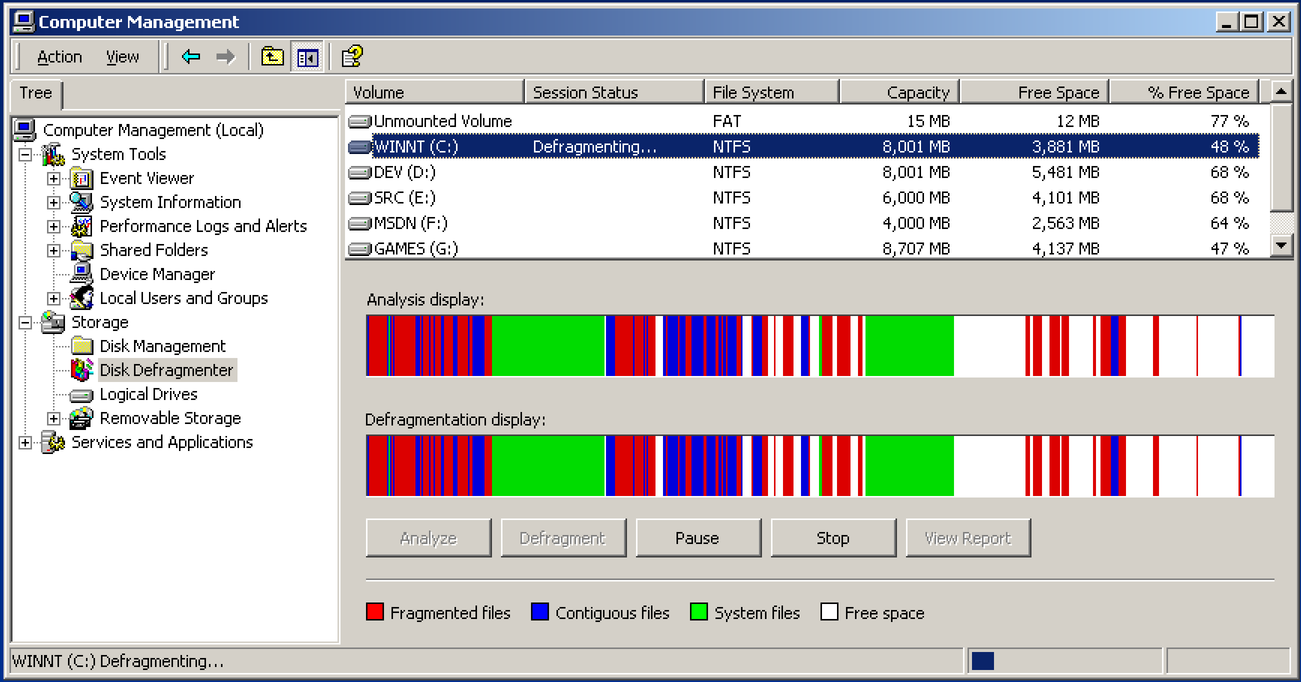
- Fragmentation: A file is fragmented if its data occupies discontiguous clusters





Defragmentation

- A common myth is that NTFS doesn't fragment, but it does
  - Defragmentation APIs have been present since NT 4.0
  - Windows 2000 introduced a non-schedulable graphical defragmenter
  - A command line interface was added in Windows XP



Volume	Session Status	File System	Capacity	Free Space	% Free Space
Unmounted Volume		FAT	15 MB	12 MB	77 %
WINNT (C:)	Defragmenting...	NTFS	8,001 MB	3,881 MB	48 %
DEV (D:)		NTFS	8,001 MB	5,481 MB	68 %
SRC (E:)		NTFS	6,000 MB	4,101 MB	68 %
MSDN (F:)		NTFS	4,000 MB	2,563 MB	64 %
GAMES (G:)		NTFS	8,707 MB	4,137 MB	47 %

🟥 Fragmented files 🟦 Contiguous files 🟩 System files ⬜ Free space

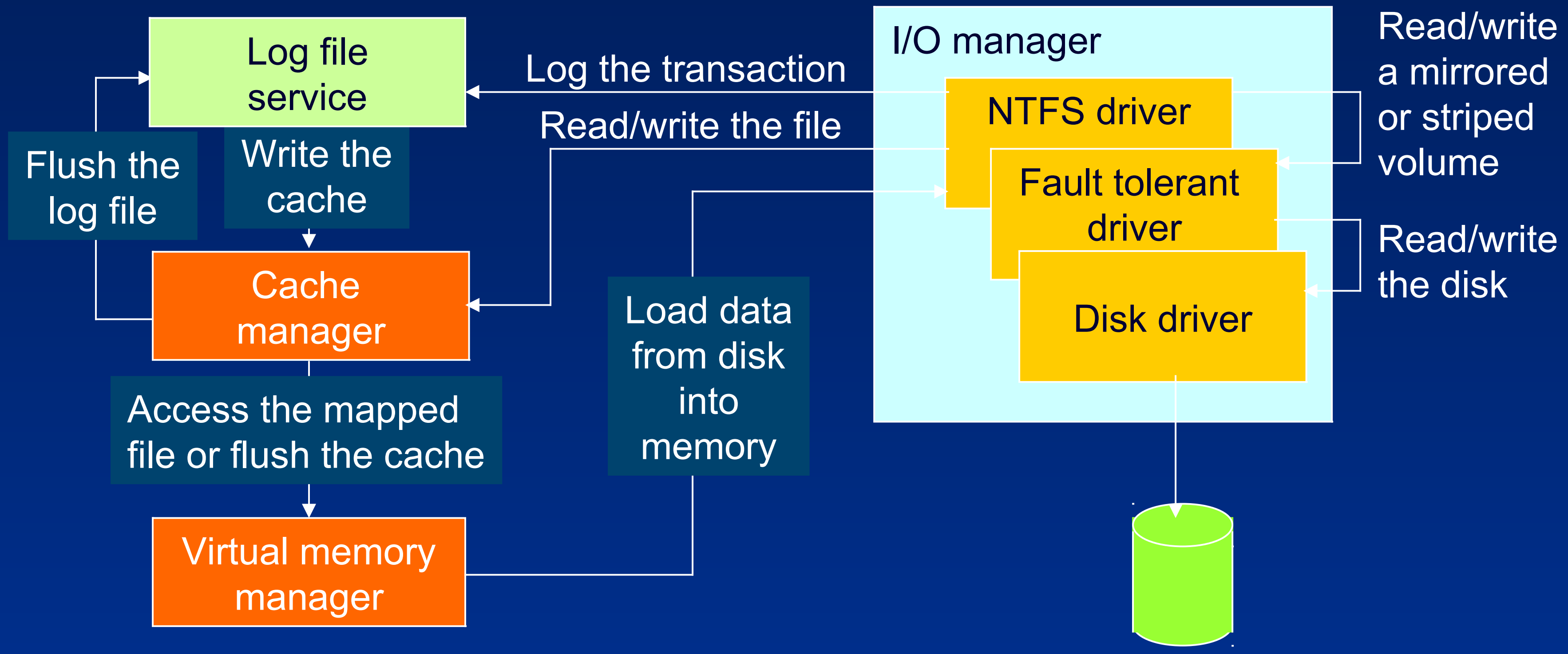
WINNT (C:) Defragmenting...

Remark: for flash-based disks (e.g. SSDs) there is no need for defragmentation !

Compression and Sparse Files

- NTFS supports transparent compression of files
  - When a directory is marked as recursively compressed, it means 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 tool shows compression ratios for compressed files)
- Sparse files are an application-controlled form of compression that define parts of a file as empty – those areas don’t occupy any disk space
  - Applications use Windows APIs to define empty areas

NTFS File System Driver



Components related to NTFS

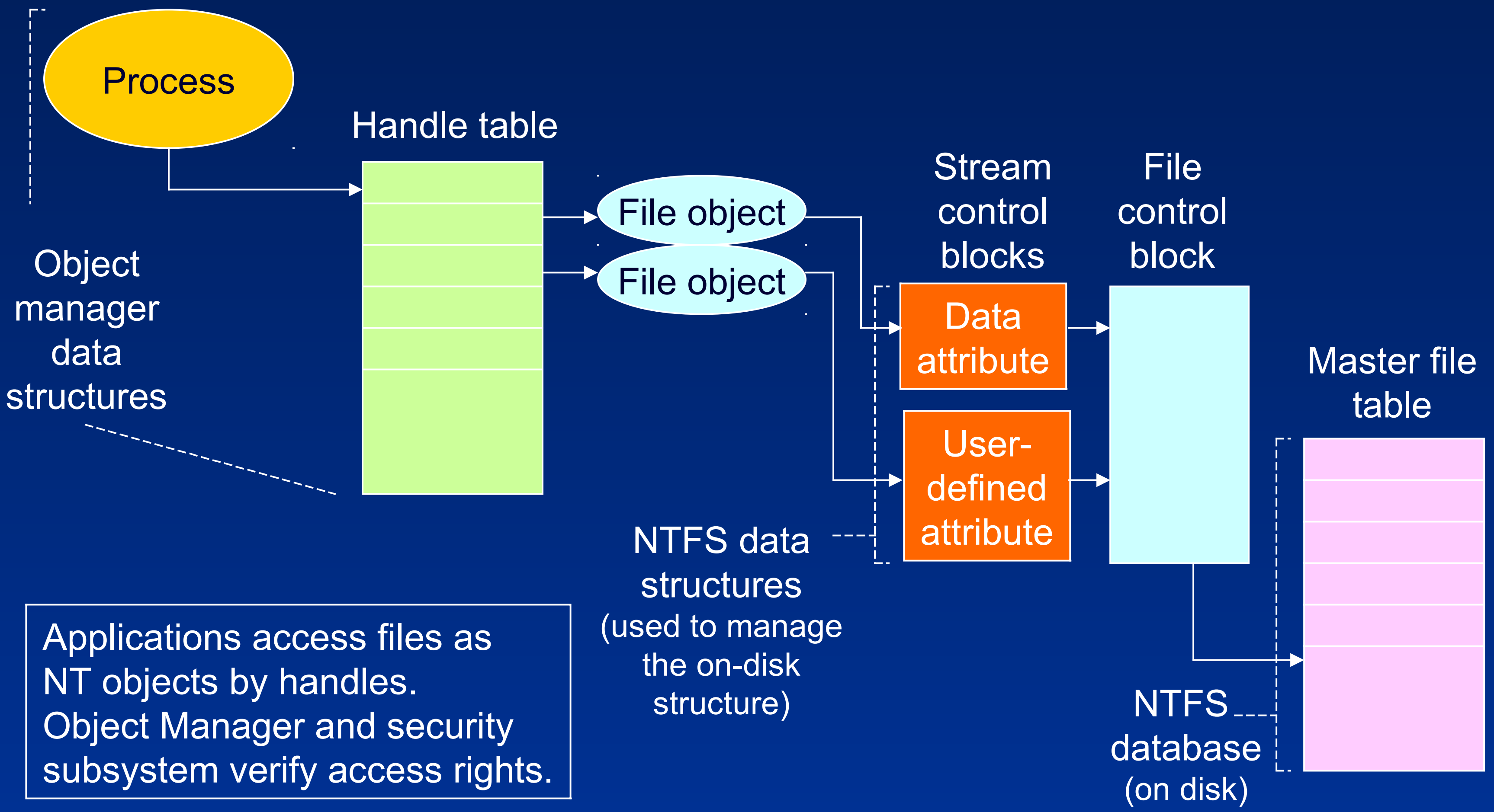
Cache Manager

- System wide caching
  - for NTFS and other file systems drivers
  - Including network file system drivers (server and redirectors)
- Cached files are mapped into virtual memory
  - Specialized interface from Cache Manager to NT virtual memory manager
  - Memory manager calls NTFS to access disk driver and obtain file

Log File Service

  - 2 copies of transaction logs
  - Transaction log is flushed to disk before write-data is sent to disk
  - Cache manager performs actual flush operation

NTFS & File Objects



Applications access files as NT objects by handles.
Object Manager and security subsystem verify access rights.

NTFS On-Disk Structure

- Volumes correspond to logical partitions on disk
- Fault tolerant volumes may span multiple disks
  - Windows 2000 introduced 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 = the number of 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

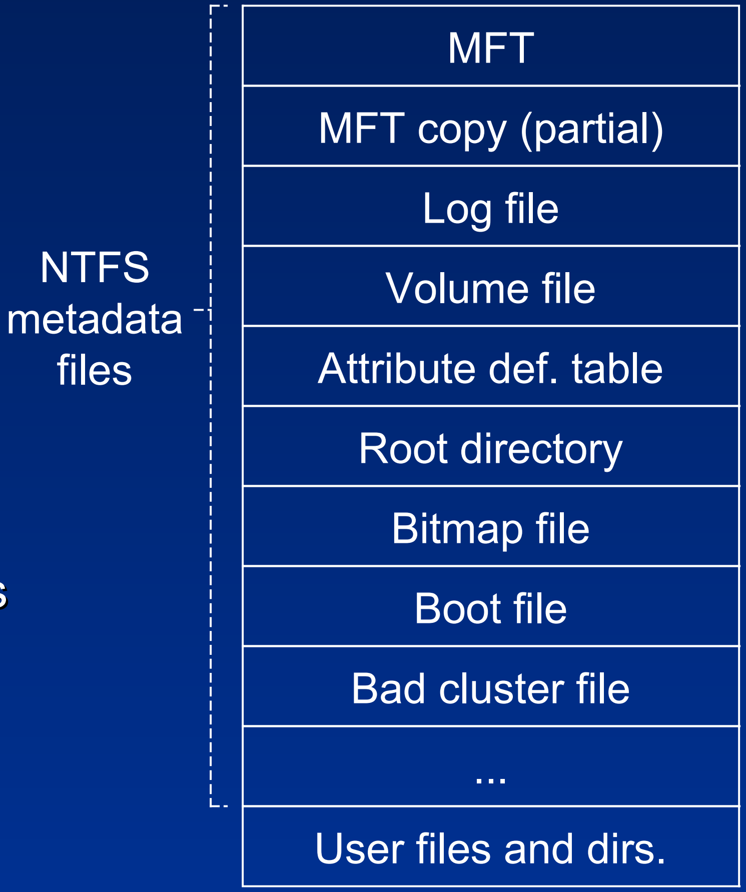
NTFS Cluster Size

- Default cluster size is disk-size dependent
  - 512 bytes for small disks (up to 512 MB)
  - 1 KB for disks up to 1 GB
  - 2 KB for disks between 1 and 2 GB
  - 4 KB for disks larger than 2 GB
- Tradeoff: disk fragmentation versus wasted space
- NTFS refers to physical locations via LCNs
  - Physical sector = LCN * cluster-factor
- Virtual Cluster Numbers (VCNs):
  - Enumerates clusters belonging to a file; mapped to LCNs
  - LCNs 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
  - Nfi.exe Utility from OEM Support Tools allows to dump MFT content (see support.microsoft.com/support/kb/articles/Q253/0/66.asp)



NTFS operation

Mounting a volume:

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

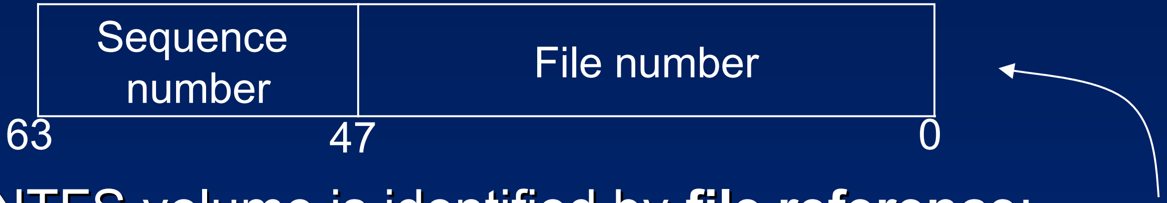
NTFS metadata

- **NTFS writes to log file** ($LogFile)
  - Record all commands that change volume structure
- **Root directory**
  - When NTFS tries to open a file, it starts the search in the root directory
  - Once the file is found, NTFS stores the file‘s MFT file reference
  - Subsequent read/write operations may access file‘s MFT record directly
- **Bitmap file** ($Bitmap)
  - Stores allocation state volume; each bit represents one cluster
- **Boot file** ($Boot)
  - Stores bootstrap code
  - Has to be located at special disk address
  - Represented as a file by NTFS → file operations are possible (!) (but no editing)

NTFS metadata (contd.)

- **Bad-cluster file** ($BadClus)
  - Records bad spots on the disk
- **Volume file** ($Volume)
  - Contains: volume name, NTFS version
  - Bit, which indicates whether volume is corrupted
- **Attribute Definition Table** ($AttrDef)
  - Defines attribute types supported on the volume
  - Indicates whether they can be indexed, recovered, etc.

File Records & File Reference Numbers

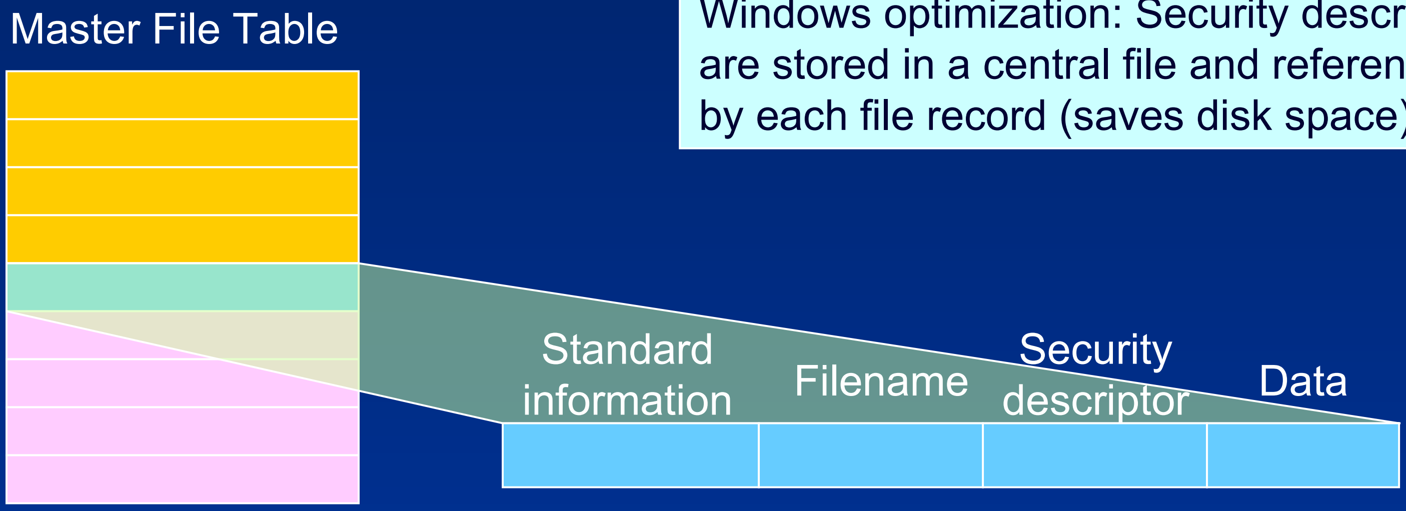


- File on NTFS volume is identified by **file reference**:
  - File number == index of file's record into the MFT
  - Sequence number – used by NTFS for consistency checking; incremented each time a reference is re-used
- File Records:
  - File is collection of attribute/value pairs (one of which is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

Windows optimization: Security descriptors are stored in a central file and referenced by each file record (saves disk space)



MFT record for a small file

Standard Attributes for NTFS Files

Attribute	Description
Standard information	File attributes: read-only, archive, etc; time stamps: creation/modification/last access time; hard link count
Filename	Name in Unicode characters; multiple filename attributes possible (POSIX links!!); short names for access by MS-DOS and 16-bit Win9x 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 attribute
Index root, Index allocation, Bitmap	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 (only for files which require multiple MFT file records for storing all of file's attributes)

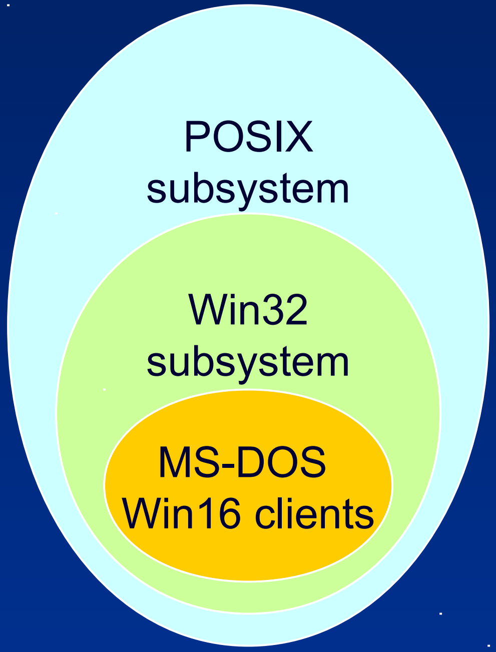
Attributes (contd.)

- 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 a MFT record are ordered by typecodes
  - Some attribute types may appear more than once (e.g. Filename)

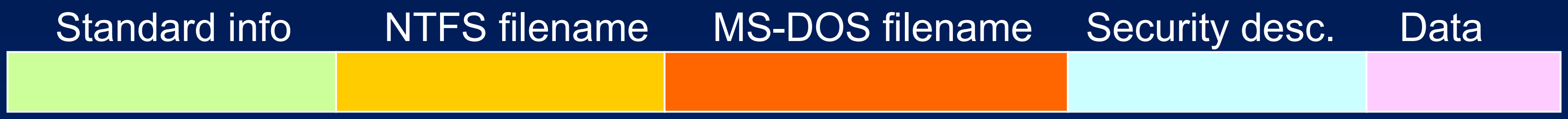
Filenames

- POSIX:
  - Case-sensitive, trailing periods & spaces
  - NTFS namespace is equivalent to POSIX space
- Win32:
  - Long filenames, Unicode names
  - Multiple dots, embedded spaces, beginning dots
- MS-DOS:
  - 8.3 names, case does not matter
- NTFS generates MS-DOS names for Win32 files automatically
  - Fully functional aliases for NTFS names
  - Stored in same directory as long names; dir /x

Namespaces



MS-DOS filenames in NTFS

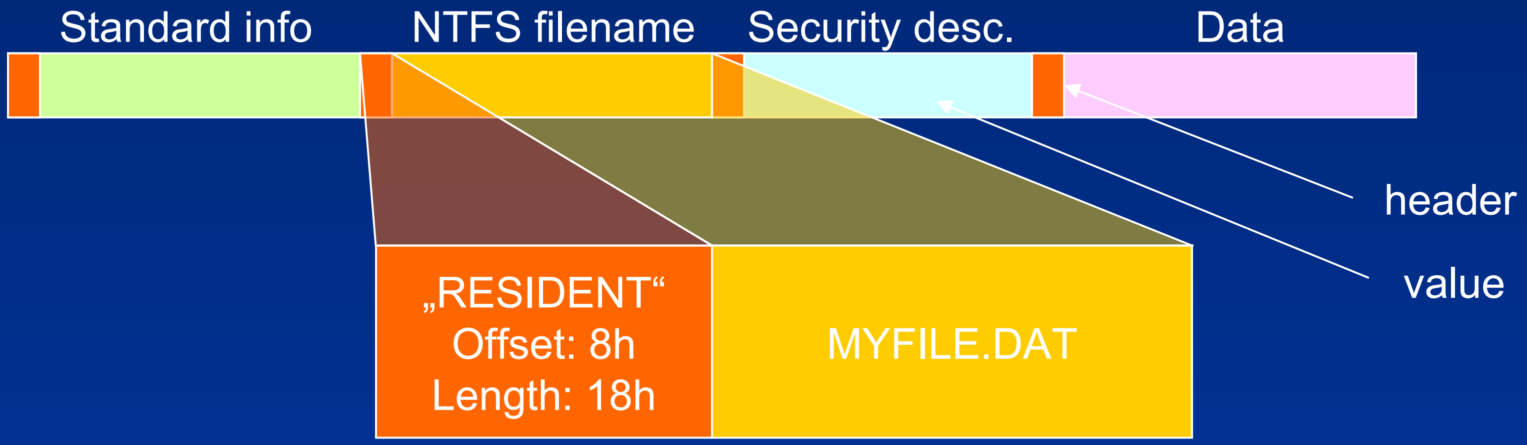


MFT file record with MS-DOS filename attribute

- NTFS name and MS-DOS name are stored in same file record and refer to same file
  - Renaming changes both filenames
  - Open, read, write, delete work with both names equally
- POSIX hardlinks are implemented in similar way
  - Deleting a file with multiple names only decreases link count
- Generation of MS-DOS names:
  1. Remove all illegal chars; remove all but one period; truncate to 6 chars
  2. Append ~1 to name; truncate extension to 3 chars; all uppercase
  3. Increment ~1 if filename duplicates an existing name in directory

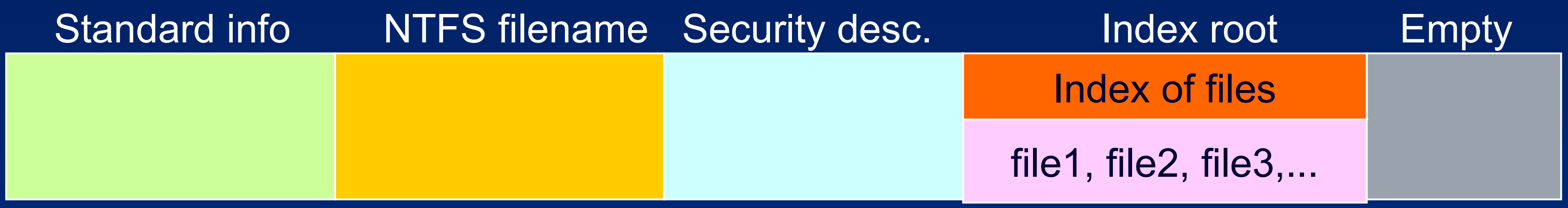
Resident & Nonresident Attributes

- Small files:
  - All attributes and values fit into file’s MFT record
  - An attribute with its value stored in file’s MFT record 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 all the files and subdirectories from that directory

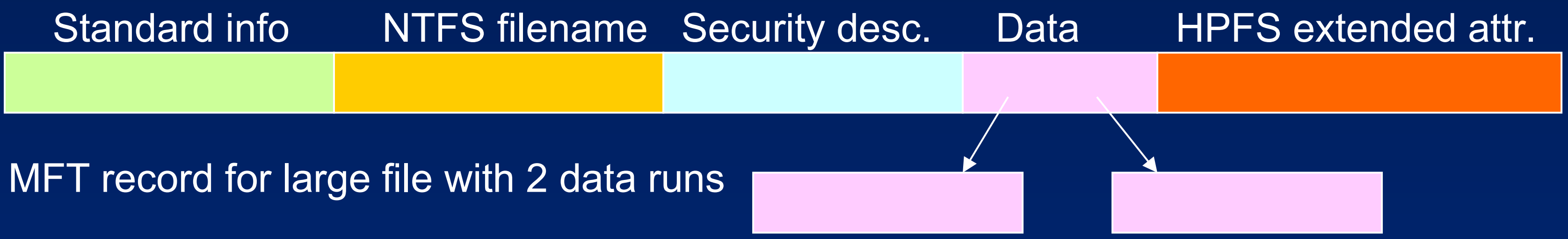


MFT file record for a small directory

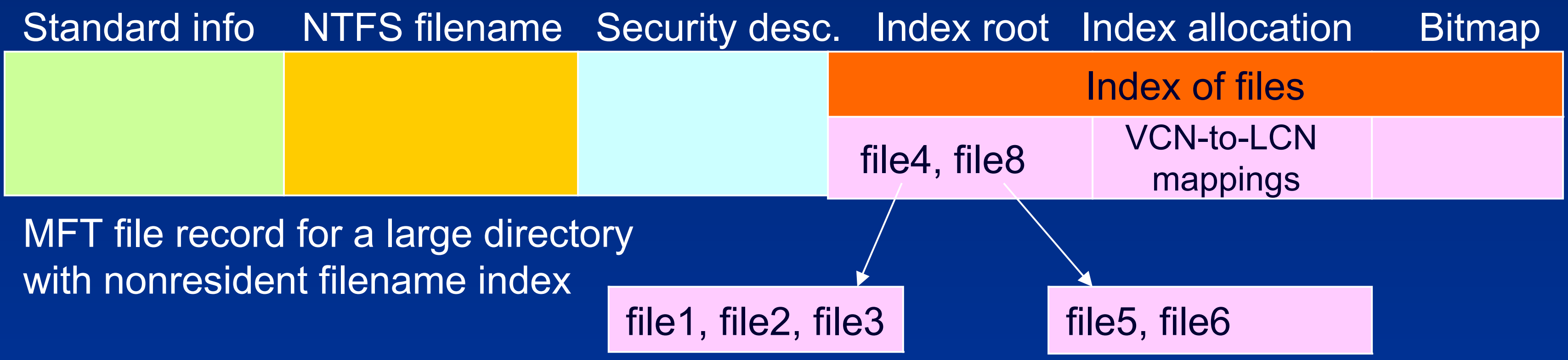
If a file's attribute does not fit into file's MFT record:

- NTFS allocates separate cluster(s) to store the attribute's value
- A contiguous group of clusters is called a *run* (or an *extent*)
- NTFS will allocate additional runs if an attribute's value later grows
- Those attributes are called „non-resident“
- Header of non-resident attribute contains location info of those runs

Large files & directories

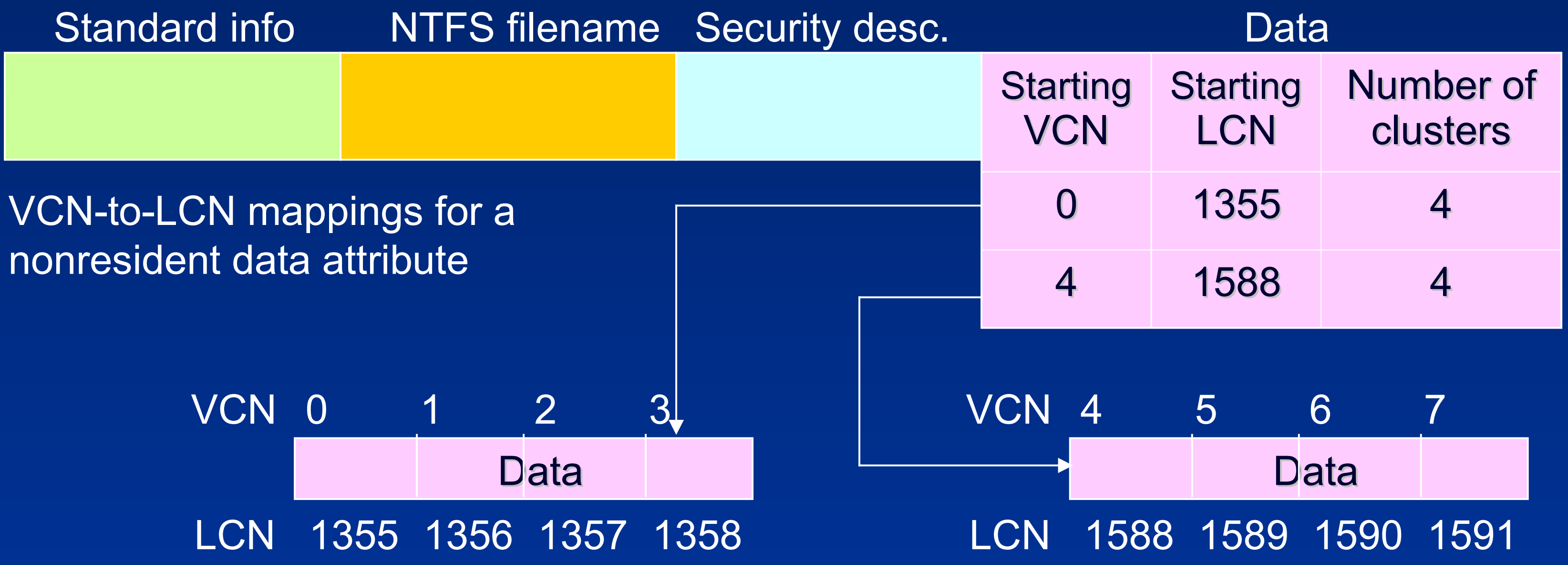


- Only attributes that can grow can be non-resident
- Filename & standard info are always resident
- Index of files for directories is represented as a B-tree



Large files (contd.)

- NTFS keeps track of runs by means of VCN-to-LCN mappings
  - **L**ogical **C**luster **N**umbers represent an entire volume
  - **V**irtual **C**luster **N**umbers represent clusters belonging to one file
  - Other attributes (e.g. index root or attribute lists) may extend over multiple runs, not only the data file's attribute



Starting VCN	Starting LCN	Number of clusters
0	1355	4
4	1588	4

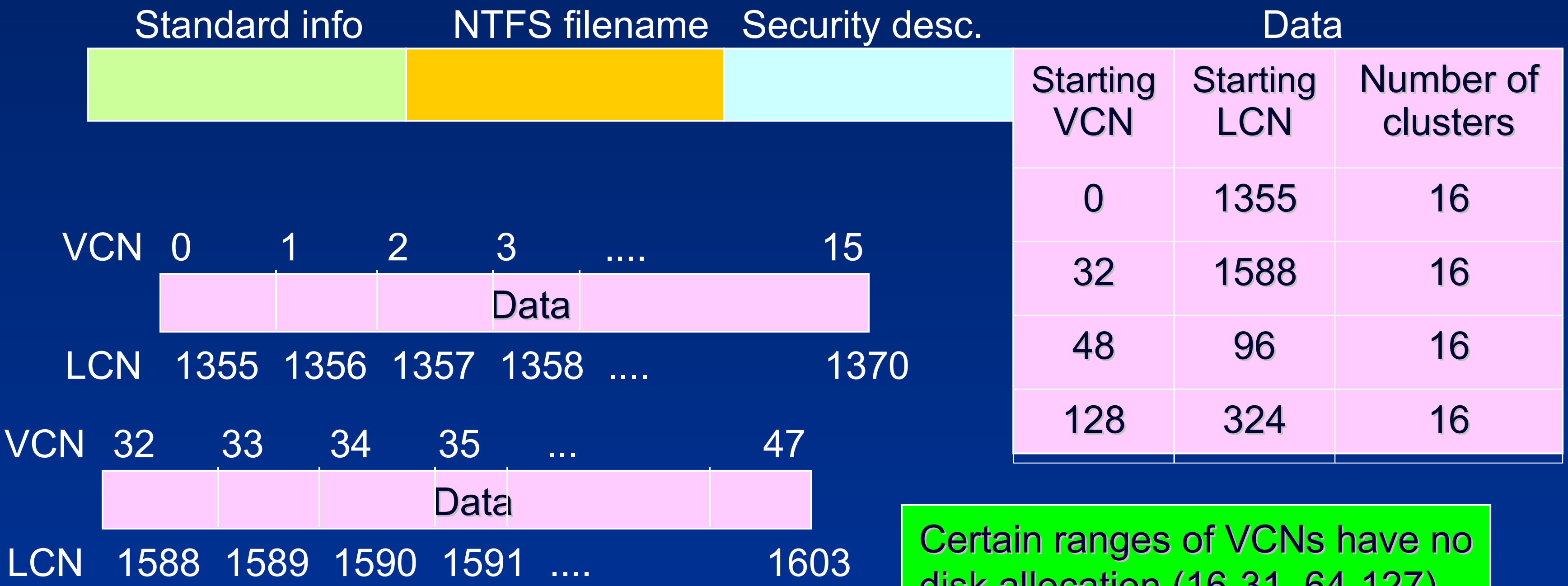
Data Compression

- NTFS supports compression
  - Per-file, per-directory, or per-volume basis
  - Using a variant of the LZ77 algorithm, known as LZNT1
  - Compression is performed on user data only, not on NTFS metadata
- Inspect compression status of volume/files via Windows API:
  GetVolumeInformation(), GetCompressedFileSize()
- Change compression setting for files/directories:
  DeviceIoControl()

  with flags
  FSCTL_GET_COMPRESSION, FSCTL_SET_COMPRESSION

Compression of sparse files

- NTFS zeroes all file contents on creation (C2 requirement)
- Many sparse files contain large amount of zero-bytes
  - These bytes occupy space on disk – unless files are compressed



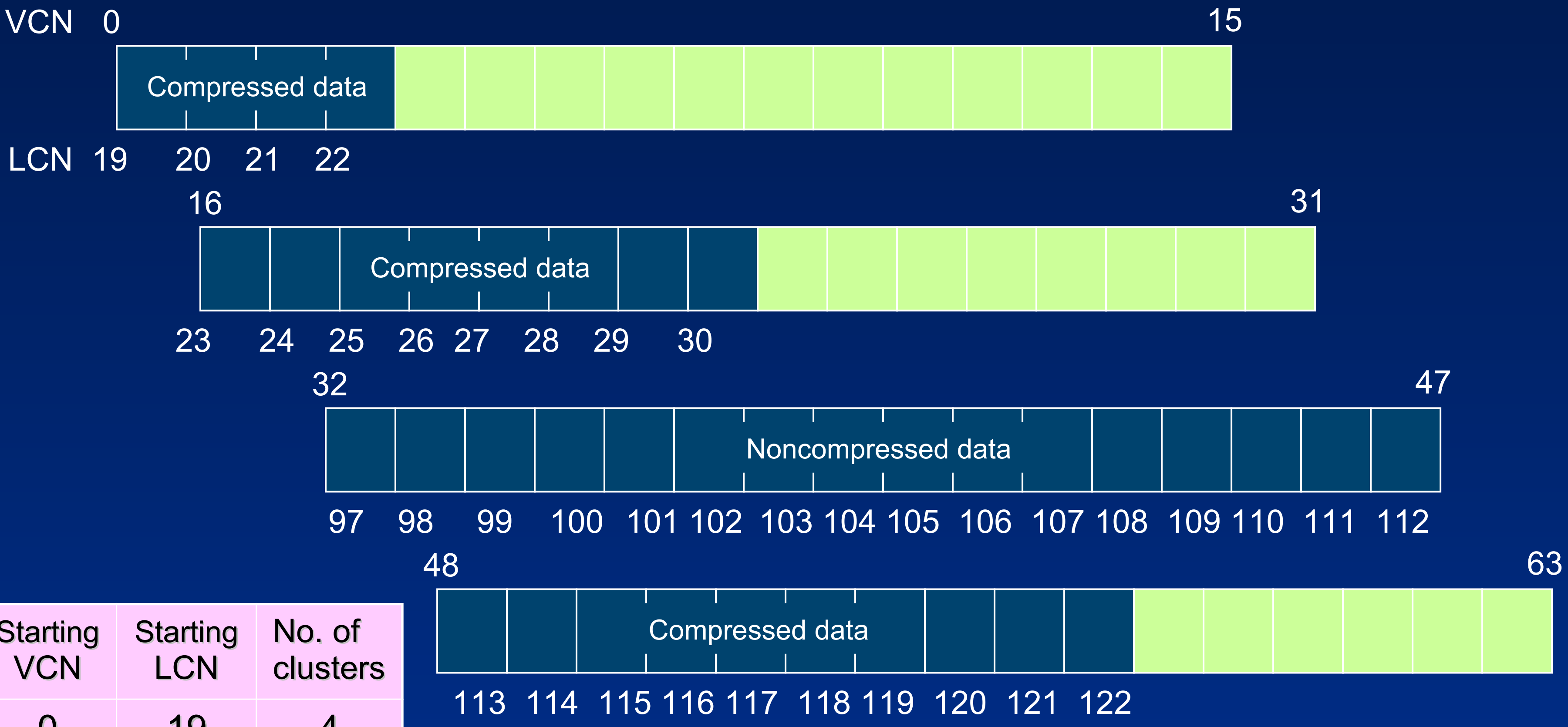
Starting VCN	Starting LCN	Number of clusters
0	1355	16
32	1588	16
48	96	16
128	324	16

Certain ranges of VCNs have no disk allocation (16-31, 64-127)

Compressing Nonsparse Data

- NTFS divides the file's unprocessed data into *compression units* which are 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 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 + asynchronous decompression improves read 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

Windows - NTFS Extensions

- Disk quotas on per-user bases
- Security descriptors (ACLs) can be stored once but referenced in multiple files
- Native support for properties (OLE), including indexing
- Reparse points – implementation of symbolic links
    - Mount points for arbitrary file system volumes
- Support for sparse files
- Distributed link tracking (via global object IDs)
    - Renaming the target file will no longer break links (shortcuts...)
- Add disk space to an NTFS volume without reboot
- No decompressing when transmitting files over network

File System Driver Architecture

- Local File System Drivers (Local FSDs):
  - Ntfs.sys, Fastfat.sys, Udfs,sys, Cdfs,sys
  - Responsible for registering with the I/O manager and volume recognition/integrity checks
  - FSD creates device objects for each mounted file system format
  - I/O manager makes connection between volume‘s device objects (created by storage device) and the FSD‘s device object
  - Local FSDs use cache manager to improve file access performance
  - Dismount operation permits the system to disconnect FSD from volume object
    - When media is changed or when application requires raw device access
    - I/O manager reinitiated volume mount operation on next access to media

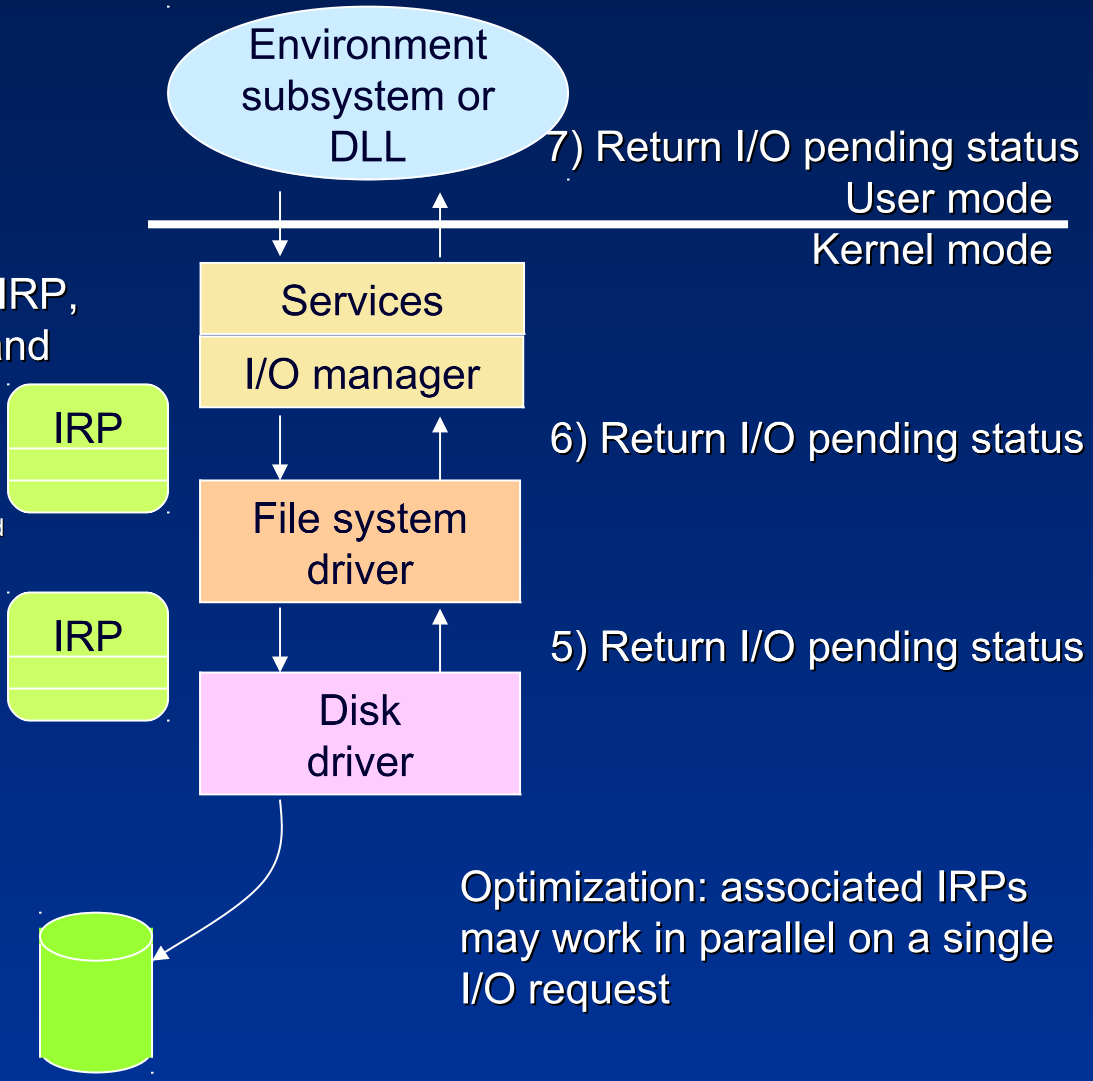
Layered Drivers - I/O System Architecture

1) Call I/O service

2) The I/O manager creates an IRP, initializes first stack location and calls file system driver

3) File system driver fills in a 2nd IRP stack location and calls the disk driver

4) Send IRP data to device (or queue IRP), and return

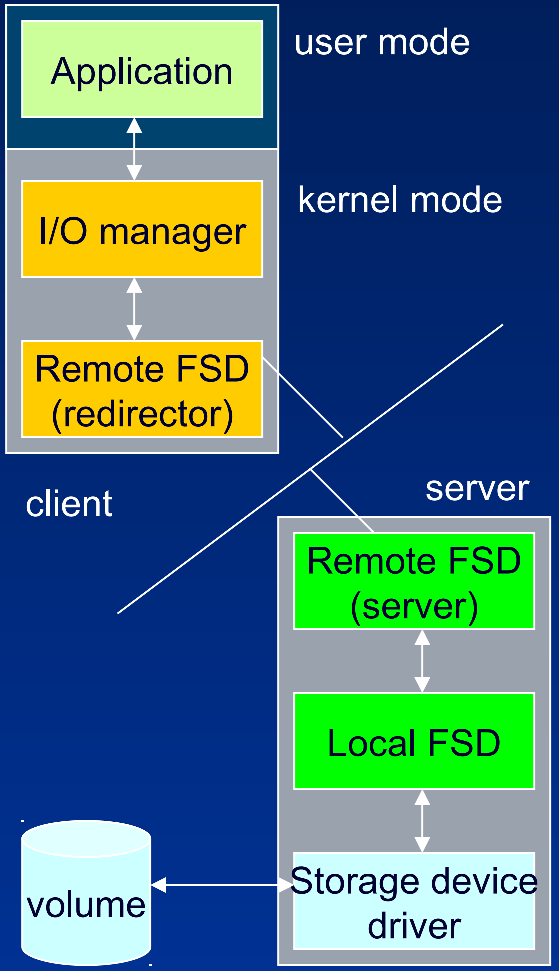

Environment subsystem or DLL
7) Return I/O pending status
User mode
Kernel mode
Services
I/O manager
IRP
6) Return I/O pending status
File system driver
IRP
5) Return I/O pending status
Disk driver

Optimization: associated IRPs may work in parallel on a single I/O request

File System Driver Architecture (contd.)

Remote File System Drivers (Remote FSDs):

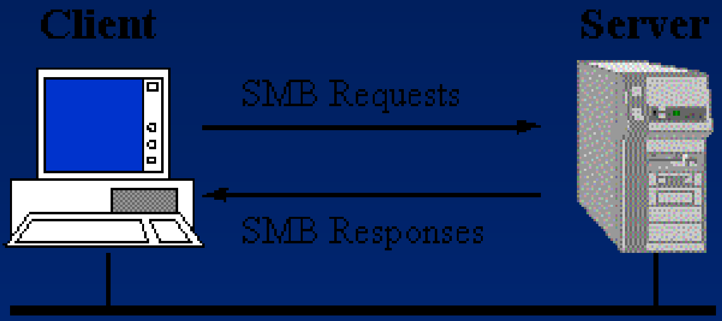
- Client-side FSD translates I/O requests from applications into network file system protocol commands
- Server-side FSD listens for network commands and issues I/O requests to local FSD
- Windows client-side remote FSD: LANMan Redirector
  - Implemented as port/miniport driver
  - Includes Windows service Workstation
- Server-side FSD server: LANMan Server
  - Includes Windows service Server
  - CIFS – Common Internet File System (an enhancement of Server Message Block protocol)



Windows Remote File Drivers: Server Message Block (SMB) protocol

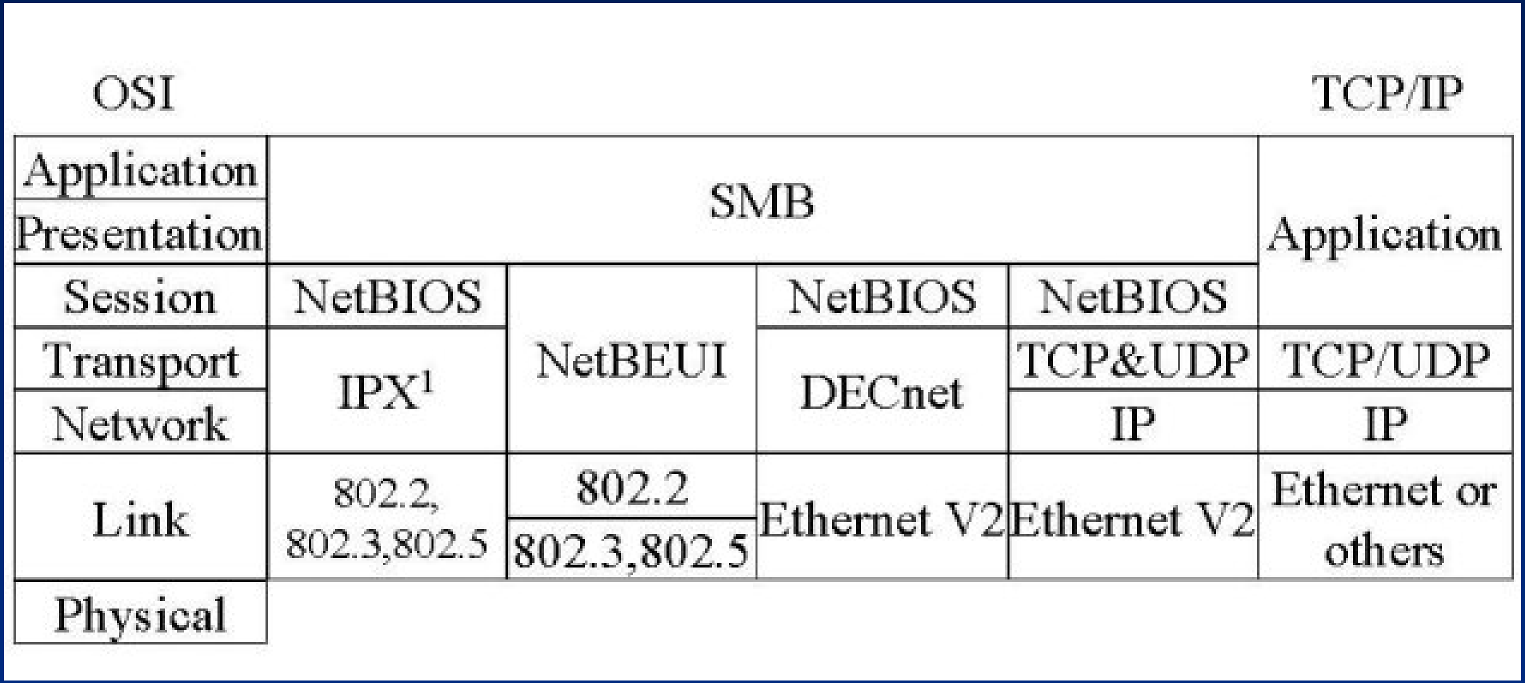
- SMB is a client/server, request-response protocol.

Additional info at http://anu.samba.org/cifs/docs/what-is-smb.html



- The only exception to the request-response nature of SMB is when the client has requested opportunistic locks (oplocks) and the server subsequently has to break an already granted oplock because another client has requested a file open with a mode that is incompatible with the granted oplock.
- In this case, the server sends an unsolicited message to the client signaling the oplock break.

SMB and the OSI model



OSI					TCP/IP
Application	SMB				Application
Presentation					
Session	NetBIOS	NetBEUI	NetBIOS	NetBIOS	
Transport	IPX[1]		DECnet	TCP&UDP	TCP/UDP
Network				IP	IP
Link	802.2, 802.3,802.5	802.2 / 802.3,802.5	Ethernet V2	Ethernet V2	Ethernet or others
Physical					

- Clients connect to servers using TCP/IP (actually NetBIOS over TCP/IP as specified in RFC1001 and RFC1002), NetBEUI or IPX/SPX.
- SMB was also sent over the DECnet protocol.
  Digital (now HP) did this for their PATHWORKS product

SMB characteristics

- NetBIOS Names
  - If SMB is used over TCP/IP, DECnet or NetBEUI, then NetBIOS names must be used in a number of cases.
  - NetBIOS names are up to 15 characters long, and are usually the name of the computer that is running NetBIOS.
  - Microsoft, and some other implementers, insist that NetBIOS names be in upper case, especially when presented to servers as the CALLED NAME.
- Protocol functionality (Core protocol):
  - connecting to and disconnecting from file and print shares
  - opening and closing files
  - opening and closing print files
  - reading and writing files
  - creating and deleting files and directories
  - searching directories
  - getting and setting file attributes
  - locking and unlocking byte ranges in files

SMB characteristics (contd.)

Security

- The SMB model defines two levels of security:
- Share level:
  - Each share can have a password, and a client only needs that password to access all files under that share.
  - This was the first security model that SMB had and is the only security model available in the Core and CorePlus protocols.
- User Level:
  - Protection is applied to individual files in each share and is based on user access rights.
  - Each user (client) must log in to the server and be authenticated by the server.
  - When it is authenticated, the client is given a UID which it must present on all subsequent accesses to the server.
  - This model has been available since LAN Manager 1.0.

SMB Clients and Servers

- Clients:
  - Included in WfW 3.x, Win 9x, and Windows NT / 2000 /…/ Win10
  - smbclient from Samba, smbfs for Linux, SMBlib
- Servers:
  - Microsoft Windows for Workgroups 3.x, Win9x, and Windows NT / 2000 /…/ Win10
  - Samba, for several UNIXes (Linux, Solaris, SunOS, HP-UX, ULTRIX, DEC OSF/1, Digital UNIX, Dynix (Sequent), IRIX (SGI), SCO Open Server, DG-UX, UNIXWARE, AIX, BSDI, NetBSD, NEXTSTEP, A/UX)
  - The PATHWORKS family of servers from Digital
  - LAN Manager for OS/2, SCO, etc.
  - VisionFS from SCO
  - Advanced Server for UNIX from AT&T
  - LAN Server for OS/2 from IBM

Resilient File System (ReFS)

- ReFS was introduced first in Windows Server 2012 and it is still in active development (trying to catch up with ZFS and BTRFS...)
- Design goals:
  - maximize data availability
  - scale efficiently to large data sets across diverse workloads
  - provide data integrity by means of resiliency to corruption:
    - Integrity-streams (checksums for metadata and optionally for file data)
    - Integration with Storage Spaces (Storage Spaces manages mirror or parity volumes) → automated repair of detected corruptions
    - Salvaging data (it removes corrupted data without an alternate “copy”)
    - Proactive error correction (a data integrity scanner, which periodically scans the volume, identifying latent corruptions and proactively triggering a repair of corrupt data, only if there is an alternate “copy” of that data – i.e. only for a mirror or parity volume)

More info: see https://docs.microsoft.com/en-us/windows-server/storage/refs/refs-overview

Further Reading

- Mark E. Russinovich, David A. Solomon, and Alex Ionescu, "*Windows Internals*", 6th Edition, Microsoft Press, 2012.
  - Chapter 12 – File Systems (from pp. 391)
    - File Systems supported by Windows (from pp. 392)
    - File System Driver Architecture (from pp. 398)
    - NTFS Design Goals and Features (from pp. 424)
    - NTFS On-Disk Structure (from pp. 442)

Remark: this chapter will be in part 2 of 7th edition!