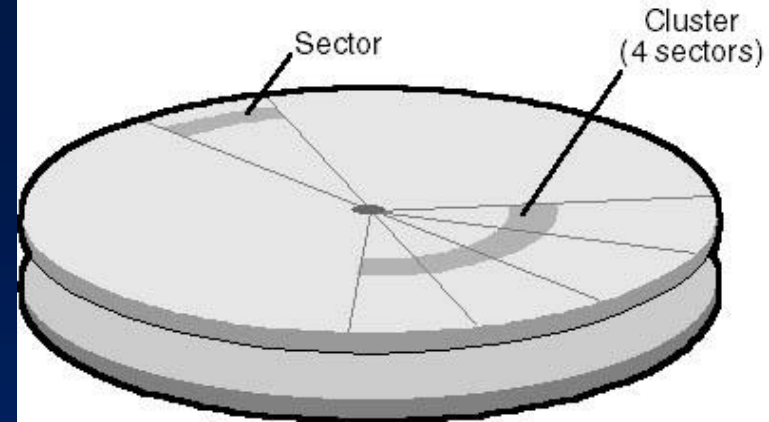


File Systems (I)

Roadmap for This Lecture

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

Windows File System Terminology



- Sectors:
 - hardware-addressable blocks on a storage medium
 - Typical sector size on hard disks for x86-based systems is 512 bytes
- 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, and FAT32)
- New Technology File System (NTFS)

CDFS

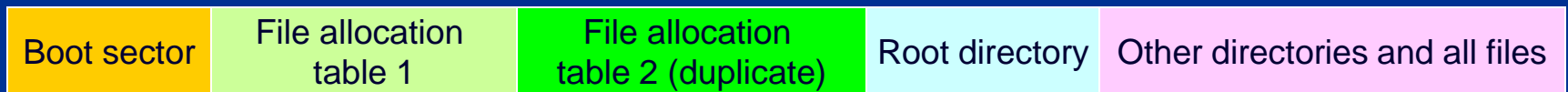
- CDFS, is a relatively simple format that was defined in 1988 as the read-only formatting standard for CD-ROM media.
- Windows 2000 implements ISO 9660-compliant CDFS in `\Winnt\System32\Drivers\Cdfs.sys`, with long filename support defined by Level 2 of the ISO 9660 standard
- Because of its simplicity, 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
 - 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, the Windows 2000 UDF driver (`\Winnt\System32\Drivers\Udfs.sys`) provides 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 a floppy disk format
- 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



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 is the most recently defined FAT-based file system format
 - it's included with Windows 95 OSR2, Windows 98, and Windows Millennium Edition
- FAT32 uses 32-bit cluster identifiers but reserves the high 4 bits, so in effect it has 28-bit cluster identifiers
 - 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-GB volumes with 512-byte clusters
- Unlike FAT12 and FAT16, root directory is not fixed size or location
 - Largest file size on Windows is 4GB (same as FAT16)

exFAT (FAT64)

- Designed for flash drives
- File size limit is 2^{64} , or 16 exa-bytes
- Max cluster size is 2^{255} sectors, in practice 32 MB
- Bitmap tracks free clusters → improves performance
- Allows 1000+ files in a single directory → scalability
- Win CE version of exFAT includes ACLs and transactions

NTFS

- NTFS is the native file system format of Windows
- NTFS uses 64-bit cluster indexes
 - Theoretical ability to address volumes of up to 16 exabytes (16 billion GB)
 - Windows limits the size of an NTFS volume to that addressable with 32-bit clusters, which is 256 TB (using 64-KB clusters)
 - Supports $2^{32}-1$ files per volume
 - Max file size is 16TB
- 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

File System Driver Architecture

- Local File System Drivers (Local FSDs):
 - Ntfs.sys, Fastfat.sys, Exfat.sys, Udfs.sys, Cdfs.sys and raw FSD
 - 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

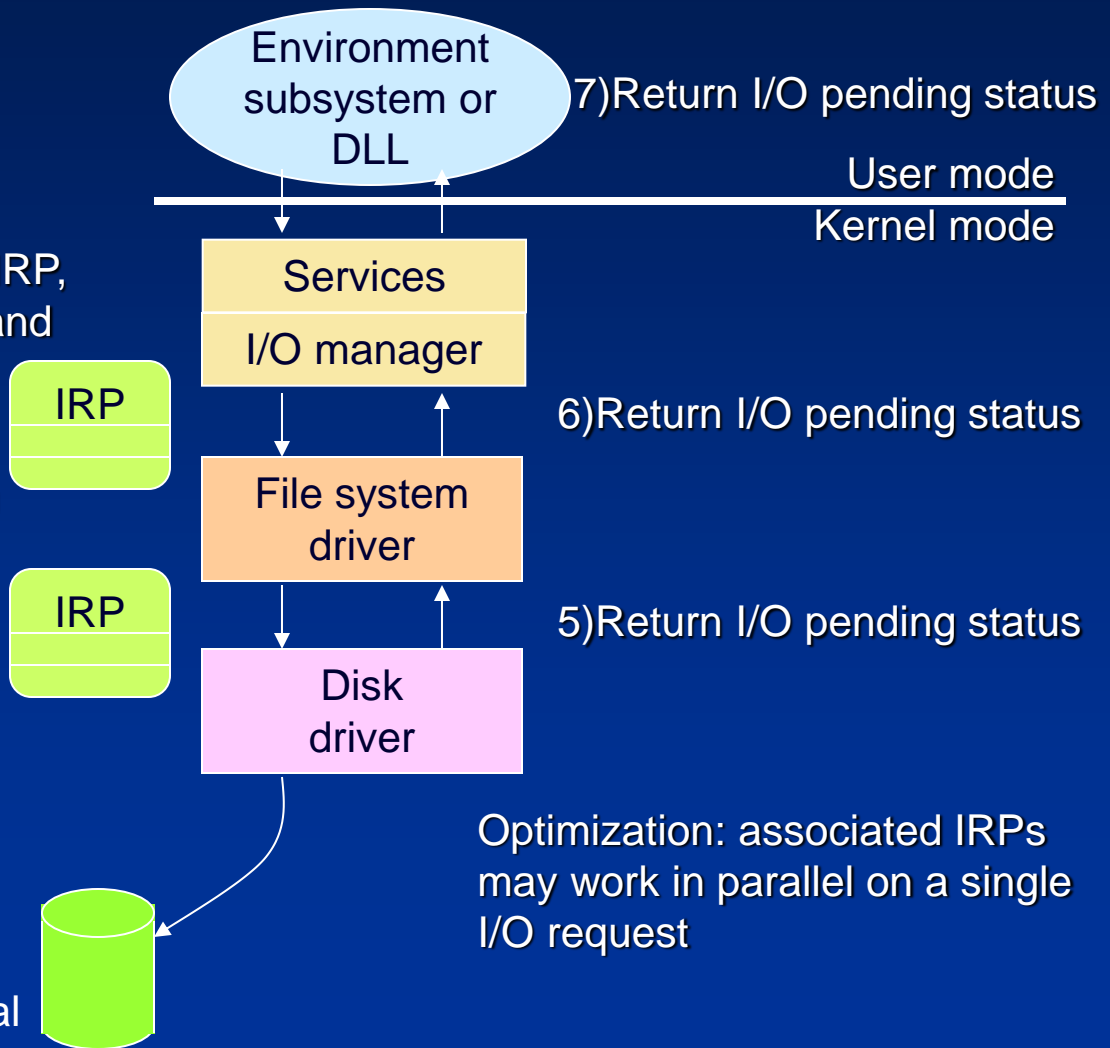
1) Call I/O service

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

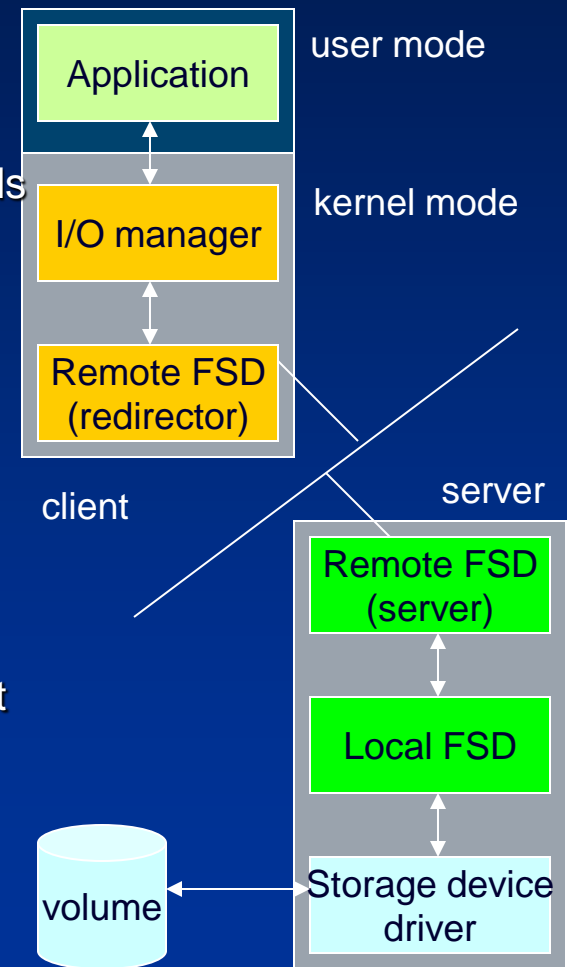
Logical volume



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 (enhancement of Server Message Block protocol)



NTFS Design Goals

- Overcome limitations inherent in FAT
 - 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

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 on-disk structures, which have single sectors containing critical file system data
 - Read error in these sectors → volume lost

NTFS Security and Redundancy

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)

Other NTFS Features

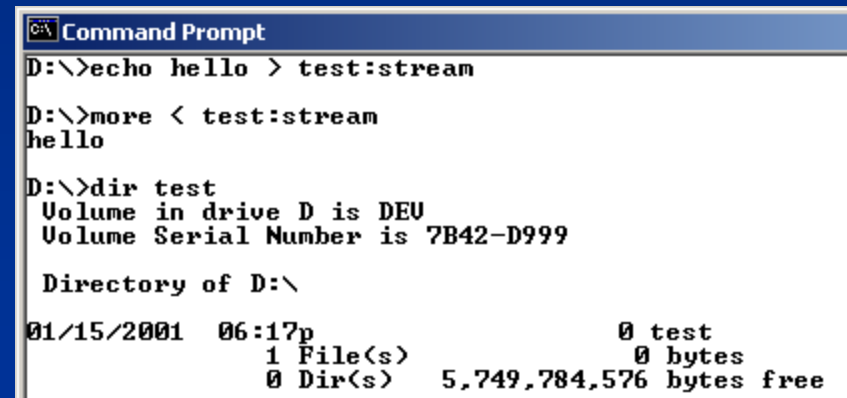
- Multiple data streams
- Unicode-based names
- Hard links
- Symbolic links and Junctions
- Compression and sparse files
- Change logging
- Per-user volume quotas
- Link tracking
- Encryption
- POSIX support
- Defragmentation
- Read-only support and dynamic partitioning

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



```
C:\> Command Prompt
D:\> echo hello > test:stream
D:\> more < test:stream
hello
D:\> dir test
Volume in drive D is DEU
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
- Directory and file names can be up to 255 chars long and contain Unicode chars, embedded spaces and multiple periods

Hard Links

- A hard link allows multiple paths to refer to the same file (doesn't support directories)
 - 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 *In* POSIX function
 - Tools: *fsutil hardlink create* or *mklink /H*
 - On-disk local refs →
can't span Disks or volumes

```
D:\>echo hello > test.txt
D:\>mklink hard.txt test.txt /H
Hardlink created for hard.txt <<===>> test.txt
D:\>dir *.txt
Volume in drive D has no label.
Volume Serial Number is 86AC-06A7

Directory of D:\

12/26/2010  11:28 AM                8 hard.txt
12/26/2010  11:28 AM                8 test.txt
                2 File(s)                16 bytes
                0 Dir(s)  192,373,010,432 bytes free
```

Symbolic Links

- Symbolic links (soft links), allow a directory to redirect file or directory pathname translation to an alternate directory
 - If the path `C:\Drivers` is a symbolic link that redirects to `C:\Winnt\System32\Drivers`, an application reading `C:\Drivers\Ntfs.sys` actually reads `C:\Winnt\System\Drivers\Ntfs.sys`
 - Symbolic links 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

Symbolic Links

- You can create symlinks with mklink command:
- Symlink doesn't have file size
- Has its own modified time

```
D:\>mklink soft.txt test.txt
symbolic link created for soft.txt <<===>> test.txt

D:\>dir *.txt
Volume in drive D has no label.
Volume Serial Number is 86AC-06A7

Directory of D:\

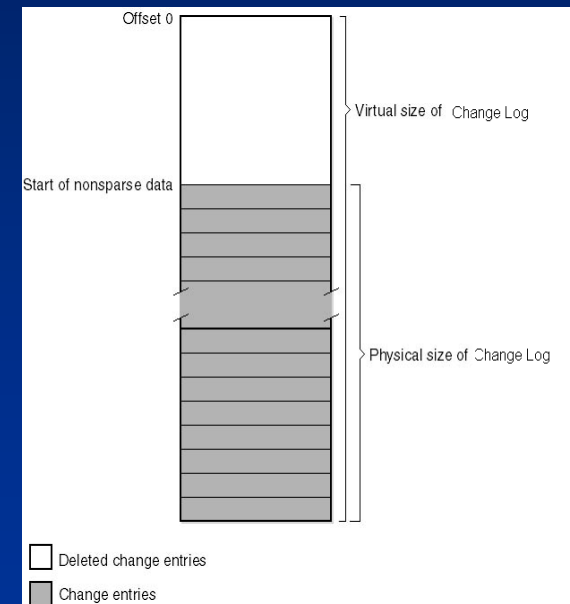
12/26/2010  11:28 AM                8 hard.txt
12/26/2010  01:39 PM          <SYMLINK>      soft.txt [test.txt]
12/26/2010  11:28 AM                8 test.txt
           3 File(s)                16 bytes
           0 Dir(s)  192,373,055,488 bytes free
```

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

- NTFS provides the change journal, which is a sparse metadata file that records file system events
 - 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 API `FSCTL_QUERY_USN_JOURNAL` to read events
 - The journal 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 than 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

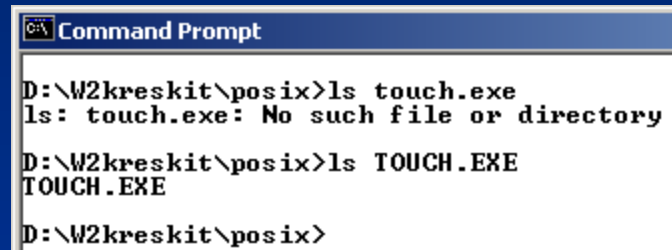
- Several types of symbolic file links are used by layered applications
 - Shell shortcuts allow users to place files in their shell namespace (on their desktop, for example) that link to files located in the file system namespace
 - Object linking and embedding (OLE) links allow documents from one application 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 volume 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 – read only access to NTFS from DOS environment
- 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 it's implemented as a file system filter driver, it's a driver that's tightly connected to NTFS

POSIX Support

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



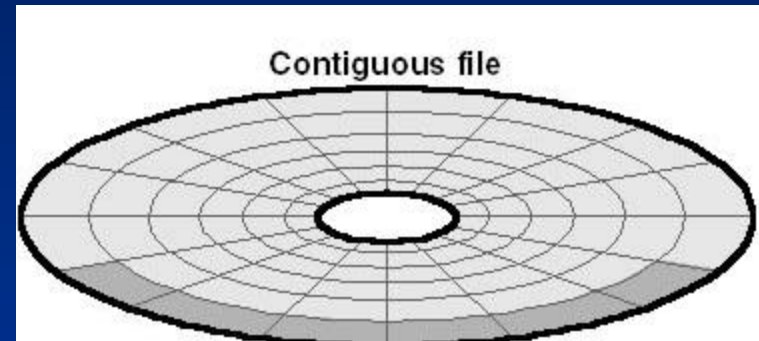
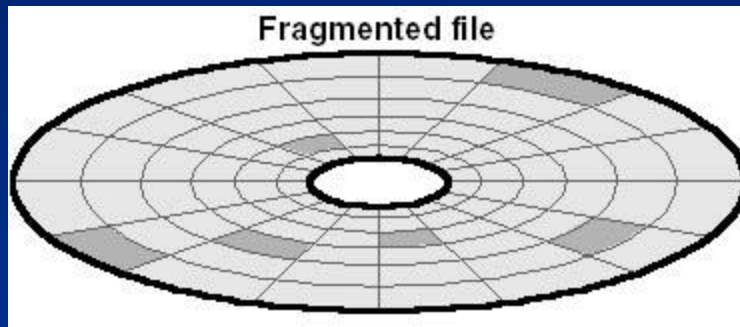
```
C:\> Command Prompt
D:\W2kreskit\posix>ls touch.exe
ls: touch.exe: No such file or directory

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

D:\W2kreskit\posix>
```

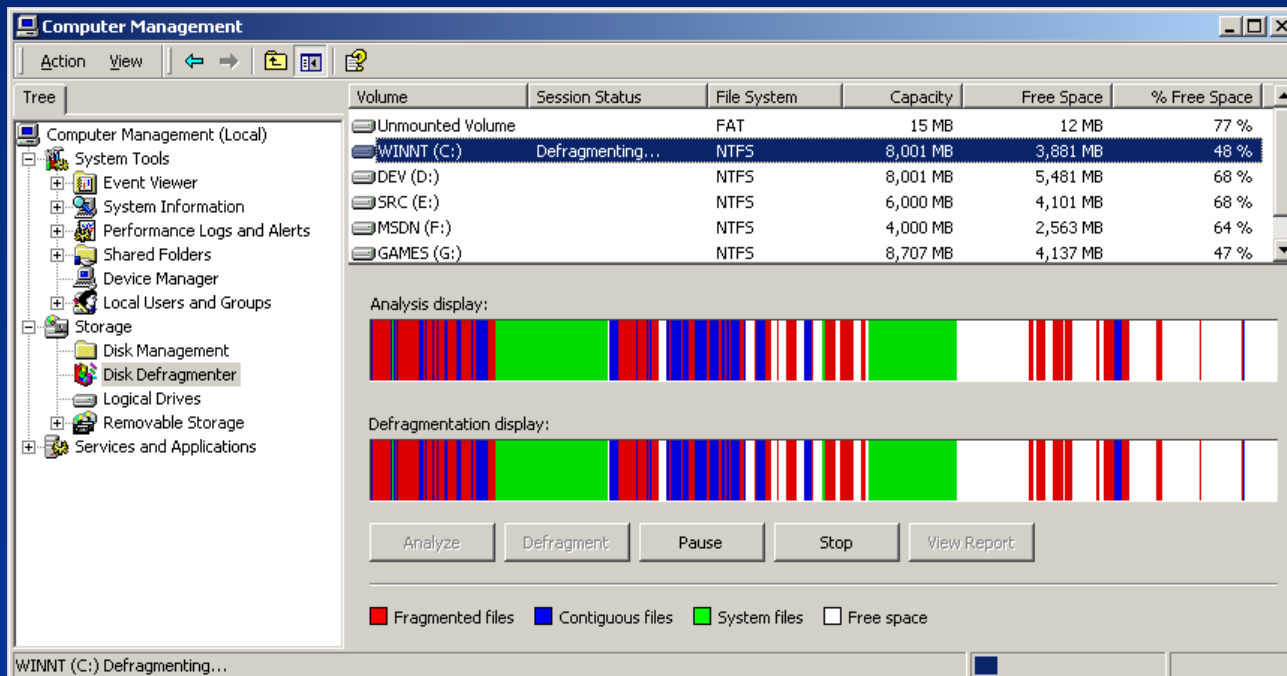
Defragmentation

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



Defragmentation

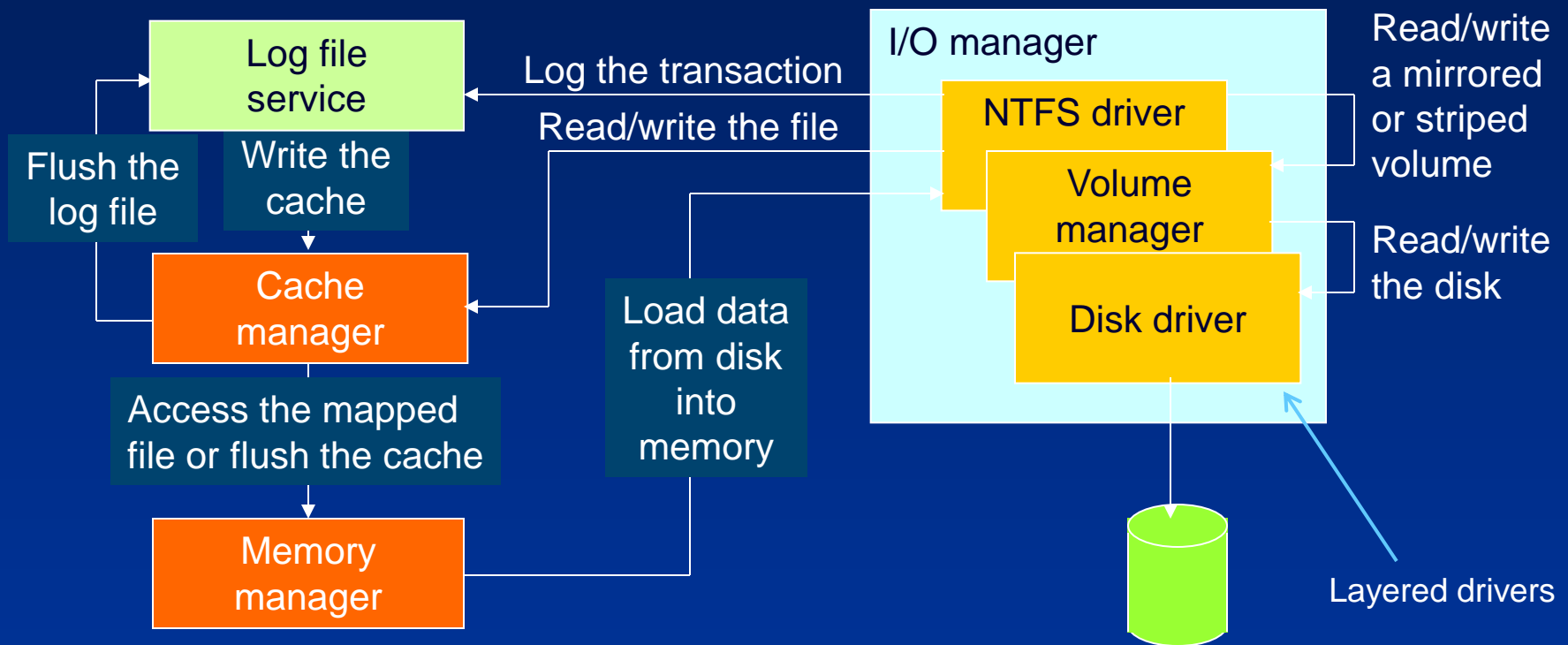
- A common myth is that NTFS doesn't fragment, but it does
 - Defragmentation APIs have been present since NT 4
 - Windows 2000 introduced a non-schedulable graphical defragmenter
 - A command line interface was added in Windows XP



Compression and Sparse Files

- NTFS supports transparent compression of files
 - When a directory is marked 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 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

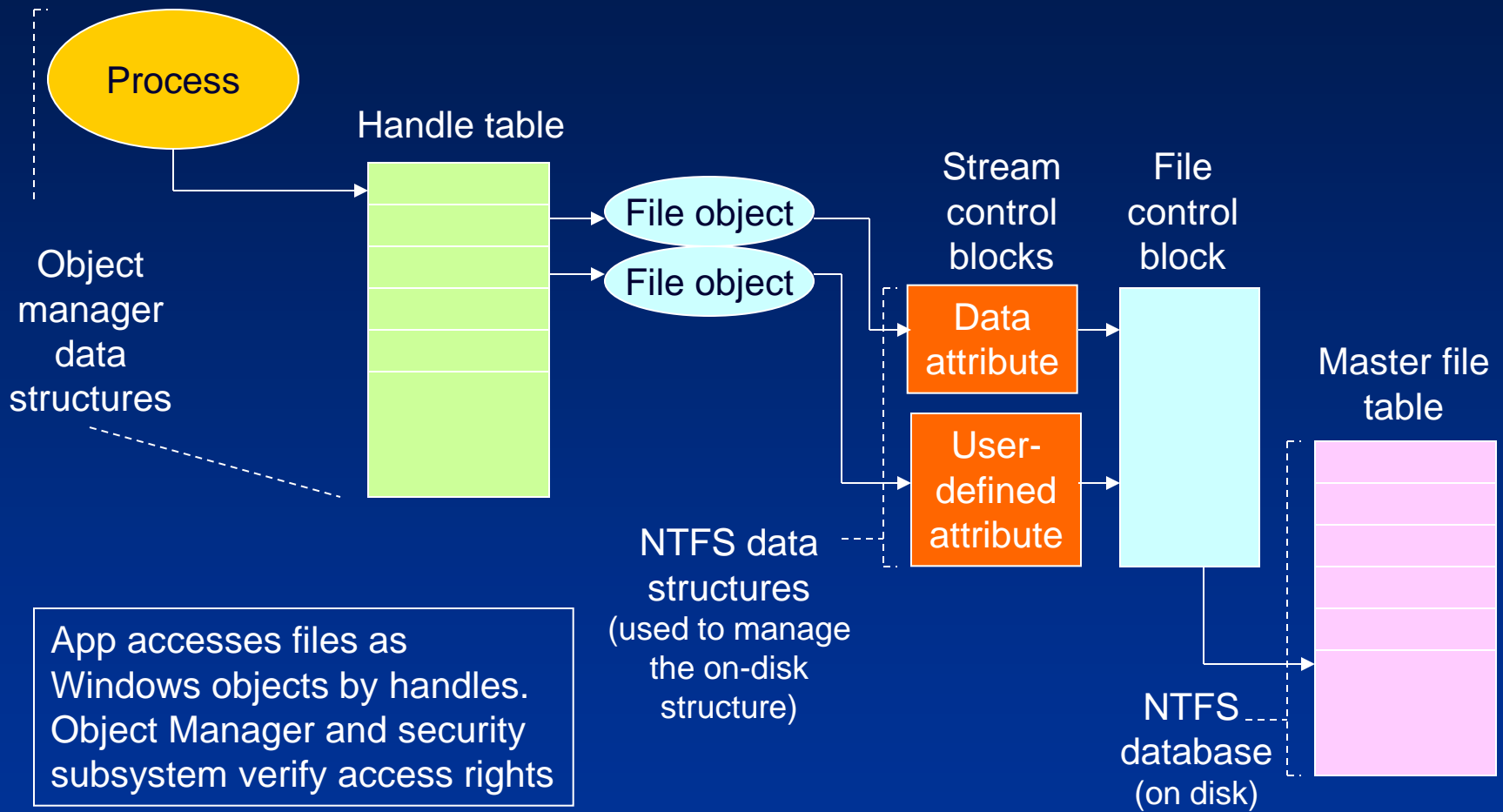
Cache Manager

- 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 file system interface from Cache Manager to Windows memory manager
 - Memory manager calls NTFS to access disk driver and obtain file

Log File Service

- Maintain a log of disk writes
- Log file used to recover NTFS volume in case of system failures
- Transaction log is flushed to disk before write-data is sent to disk
- Cache manager performs actual flush operation

NTFS & File Objects



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 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 disk address = $LCN * \text{cluster-factor}$
- 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
 - Nfi.exe Utility from OEM Support Tools allows to dump MFT content (see support.microsoft.com/support/kb/articles/Q253/0/66.asp)

NTFS
metadata
files

MFT
MFT copy (partial)
Log file
Volume file
Attribute def. table
Root directory
Bitmap file
Boot file
Bad cluster file
...
User files and dirs.

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 in case 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

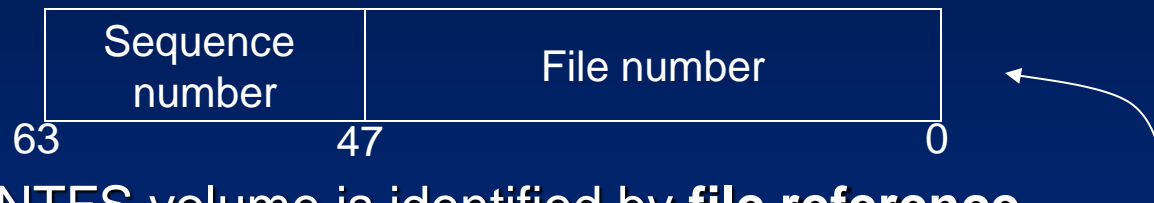
NTFS metadata

- **NTFS writes to log file (\$LogFile)**
 - Record all commands that change volume structure
- **Root directory:**
 - When NTFS tries to open a file, it starts search in the root directory
 - Once the file is found, NTFS stores the file's MFT file reference
 - Subsequent read/write ops. 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 file by NTFS → file ops. possible (!) (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

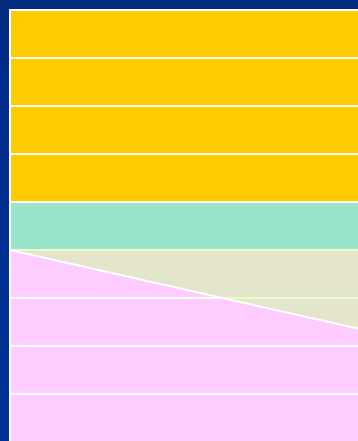


- File on NTFS volume is identified by **file reference**
 - File number == index in MFT
 - Sequence number – used by NTFS for consistency checking; incremented each time a reference is re-used
- File Records:
 - File is collection of attribute/value pairs (one of which is data)
 - Unnamed data attribute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

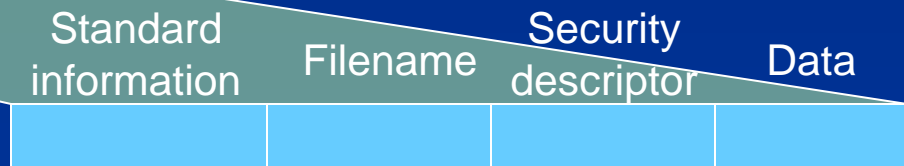
File Records (contd.)

- 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

Master File Table



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 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 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)

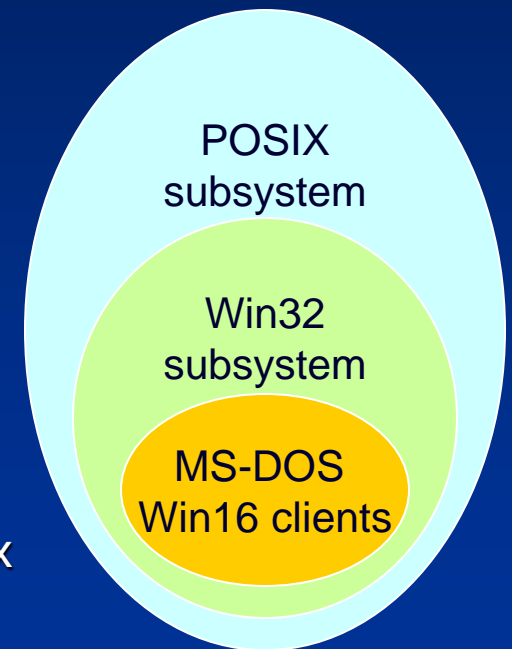
Attributes (contd.)

- 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)

Filenames

- POSIX:
 - Case-sensitive, trailing periods & spaces
 - NTFS namespace equiv. 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

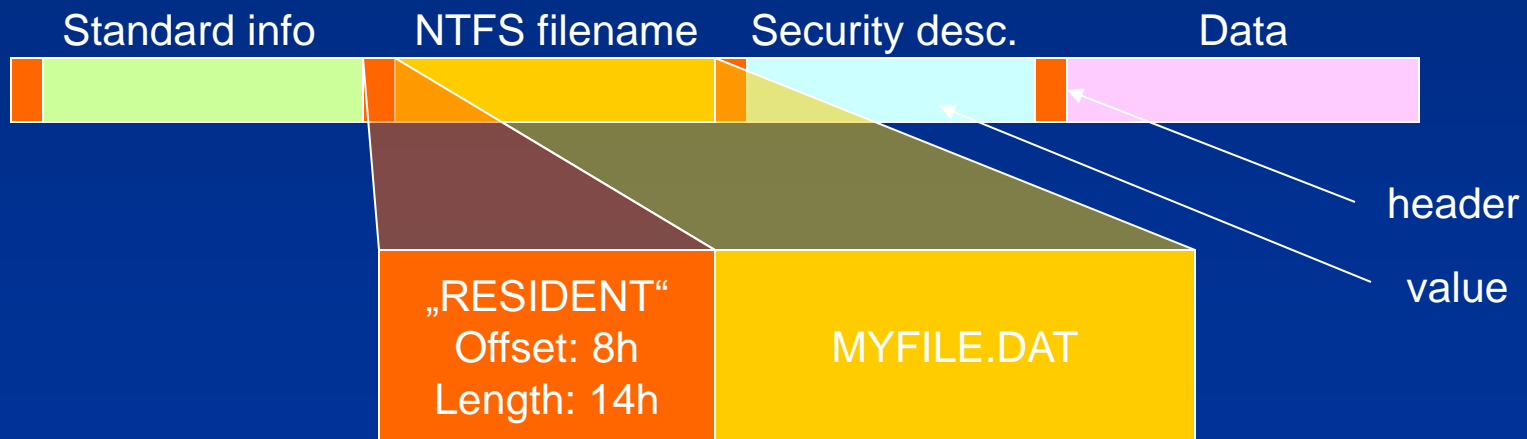


MFT file record with MS-DOS filename attribute

- 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 MFT
 - Attribute with value in MFT is called “resident”
 - All attributes start with header (always resident)
 - Header contains offset to attr. value and length of value



Attributes (cont'd.)

- Small directory:

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



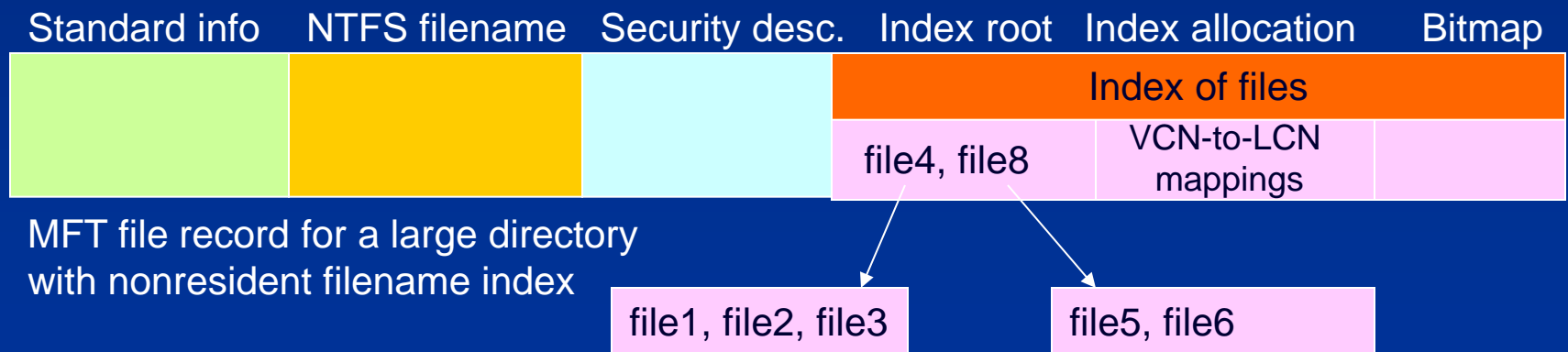
MFT file record for a small directory

- 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

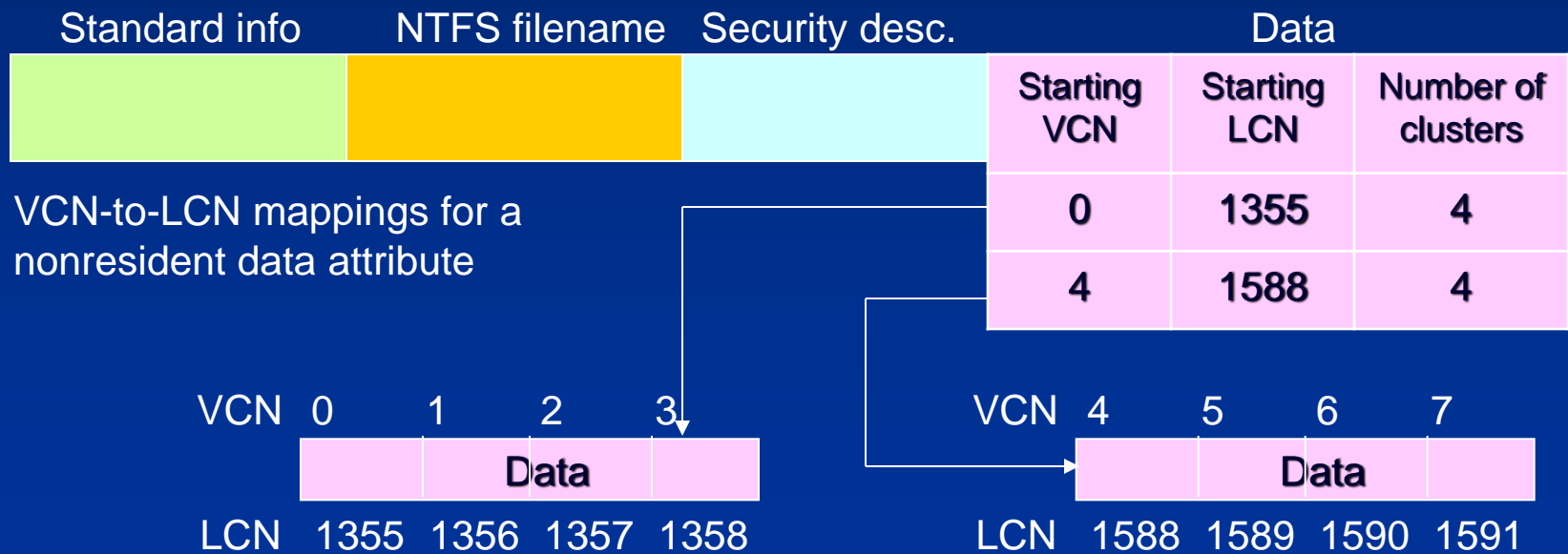


- Only attributes that can grow can be non-resident
- Filename & standard info are always resident
- Index of files for directories forms B+ tree



Large files (contd.)

- NTFS keeps track of runs by means of VCN (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

- Inspect files/volume via Winndows API:

`GetVolumeInformation(), GetCompressedFileSize()`

- Change settings for files/directories:

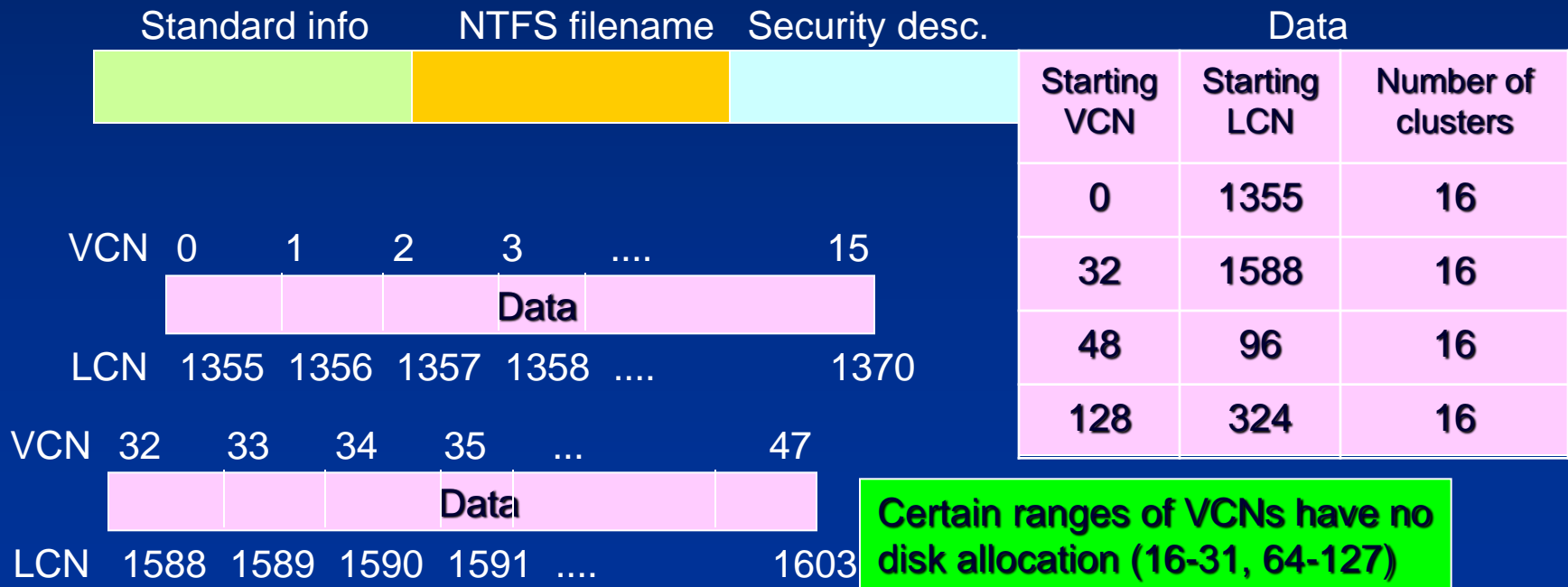
`DeviceIoControl()`

with flags

`FSCTL_GET_COMPRESSION, FSCTL_SET_COMPRESSION`

Compression of sparse files

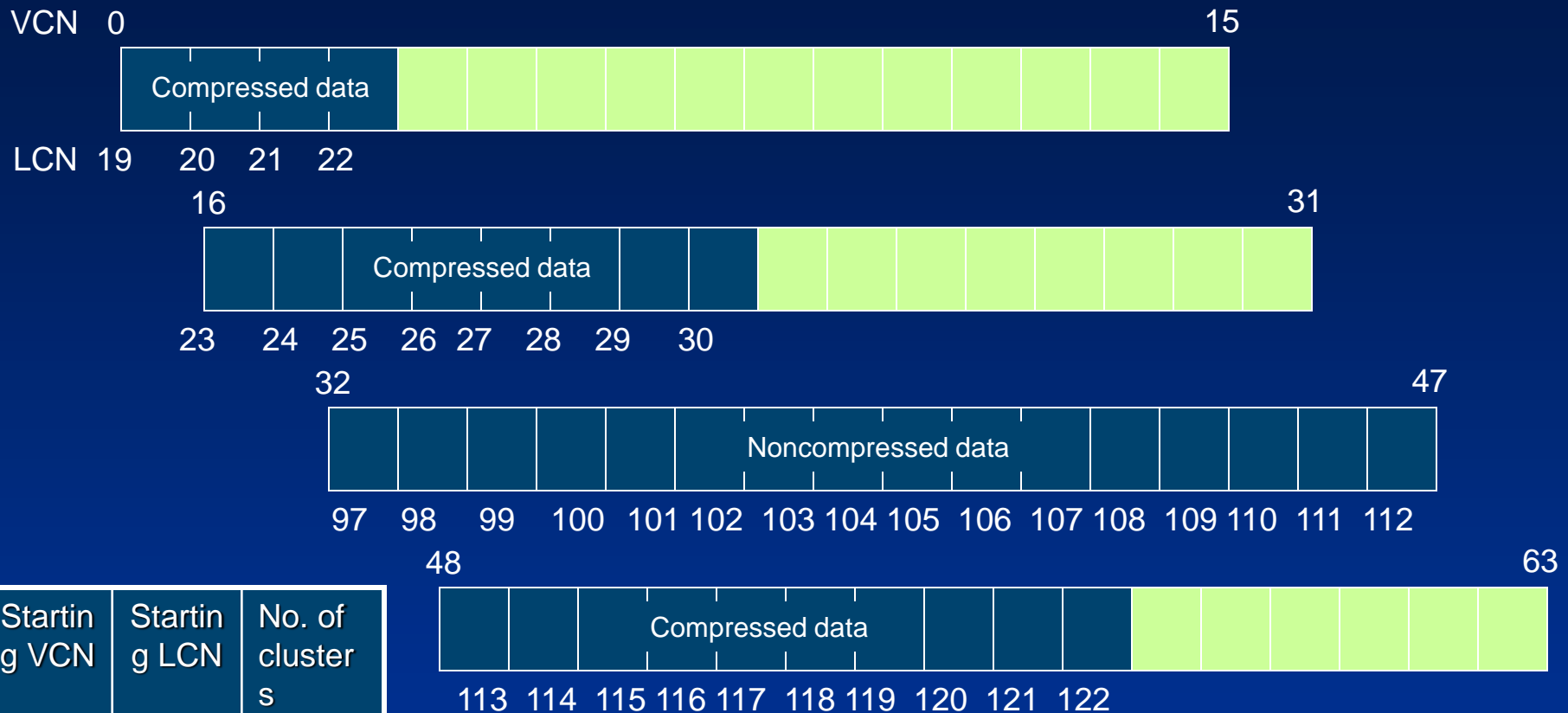
- NTFS zeroes all file contents on creation (C2 req.)
- Many sparse files contain large amount of zero-bytes
 - These bytes occupy space on disk – unless files are compressed



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 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 improves 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

Further Reading

- Mark E. Russinovich, *et al.* Windows Internals, 5th Edition, Microsoft Press, 2009.
 - File Systems supported by Windows (from pp. 890)
 - File System Driver Architecture (from pp. 895)
 - NTFS Design Goals and Features (from pp. 918)
 - NTFS On-Disk Structure (from pp. 937)

Source Code References

- Windows Research Kernel sources do not include NTFS
- A raw file system driver is included in `\base\ntos\raw`
- Also see `\base\ntos\fstrl` (File System Run-Time Library)