Mag. iur. Dr. techn. Michael Sonntag

Filesystems

Computer forensics

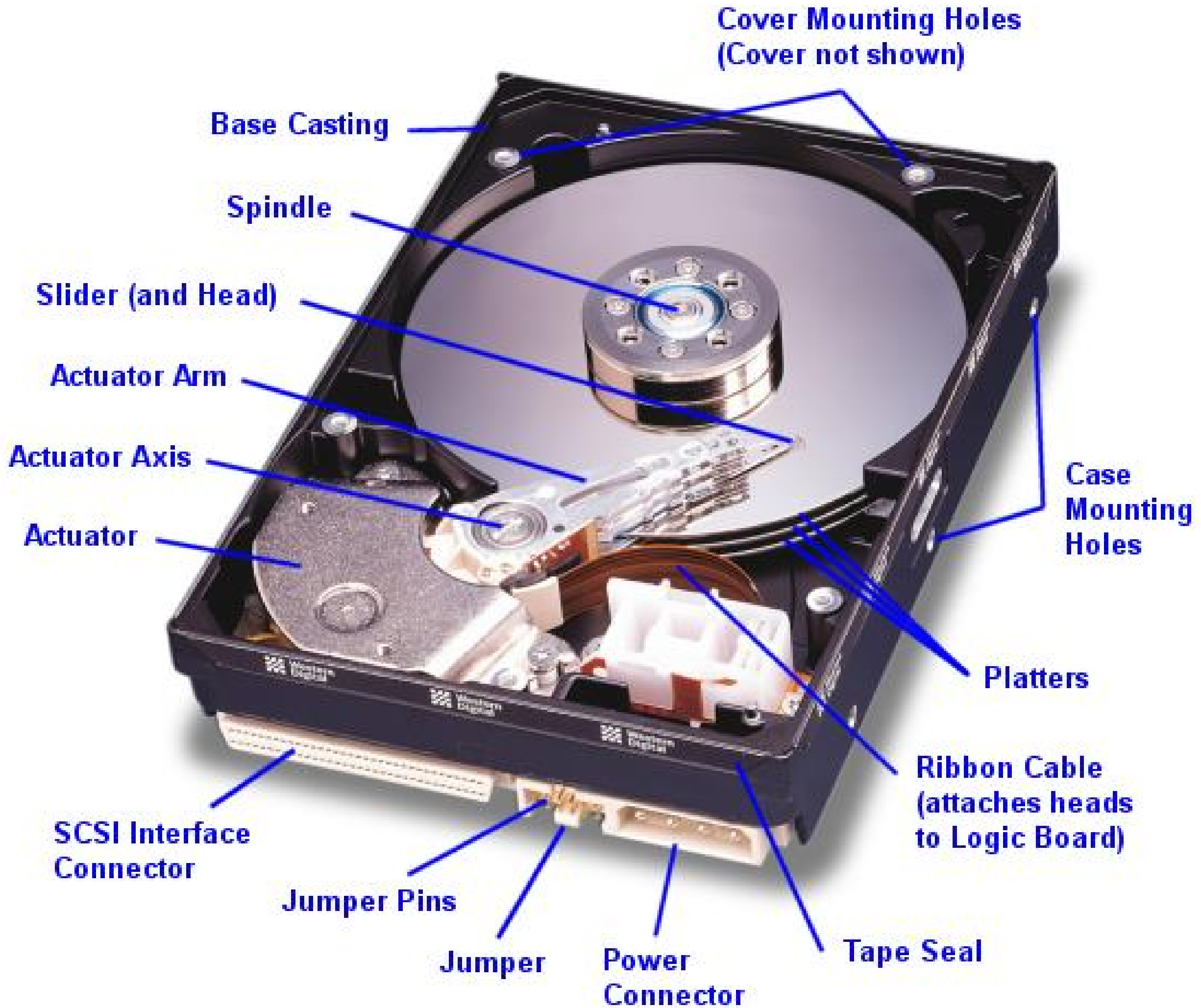
Institute for Information Processing and
Microprocessor Technology (FIM)
Johannes Kepler University Linz, Austria

E-Mail: sonntag@fim.uni-linz.ac.at
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

Agenda

- Physical disk layout
- The boot sequence
  - → What changes on a disk during a boot?
- Filesystems in detail:
  - → FAT, FAT32
  - → NTFS
  - → EXT3

Physical structure of a harddisk

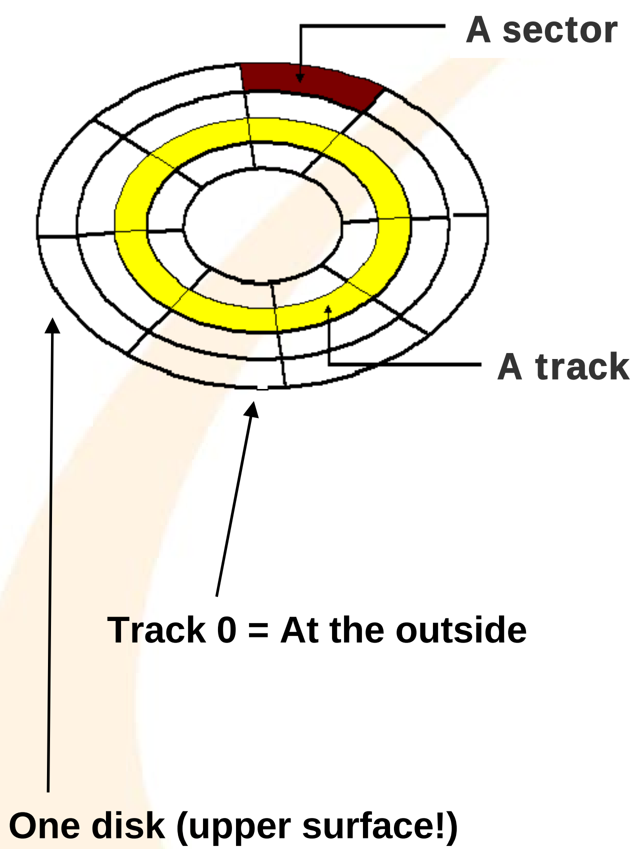


(http://www.storagereview.com/guide2000/hdd/...)

General aspects of harddisks

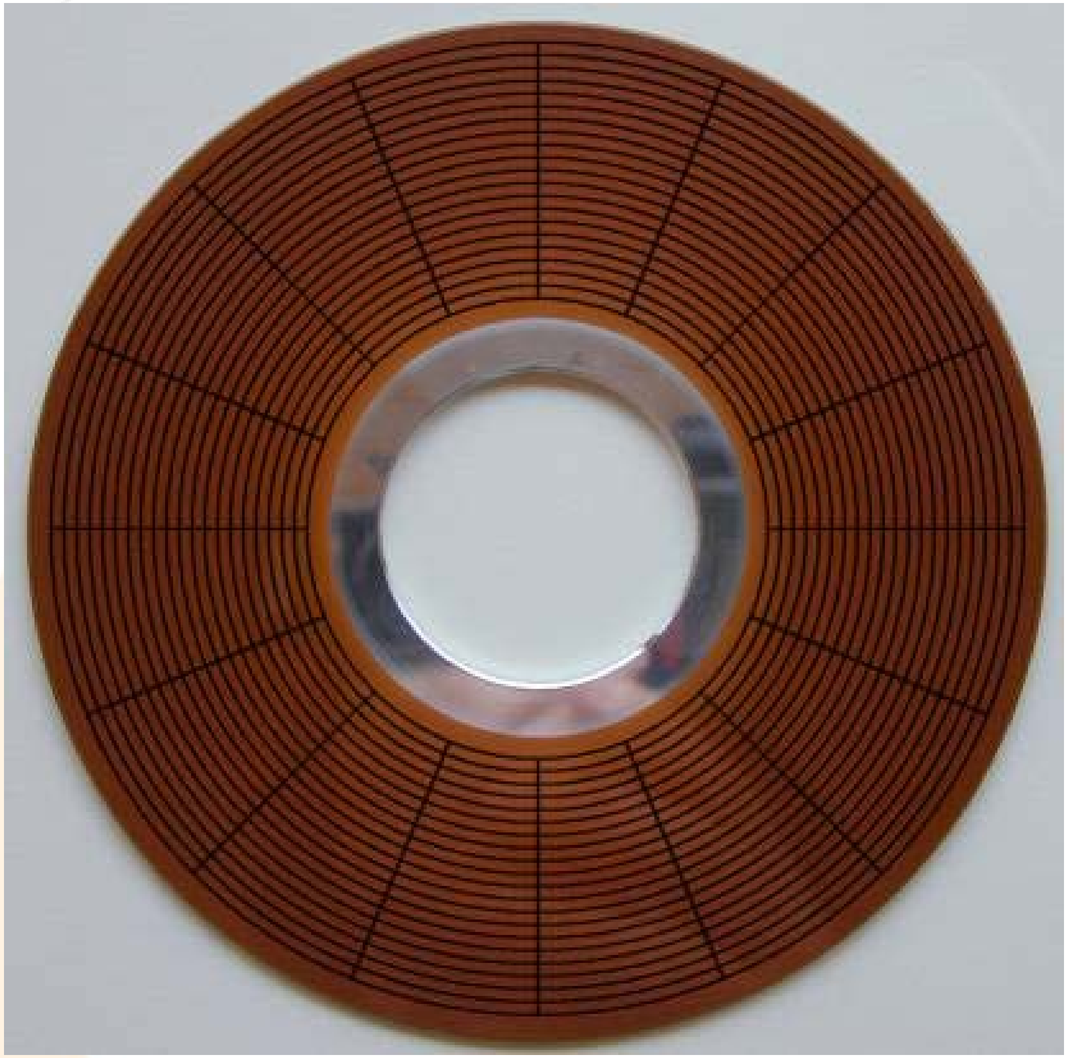
- Several different sized exist
  - → Typically named according to the size of the disks, not the case
    - » Note that these are not absolutely accurate (3,5" drive → 3,74" disk)!
- Rotating disks = „platters“
  - → Made from aluminium or compounds; perhaps even glass
  - → Coating: Ironoxide, Cobalt, …
- "Comb" with read-/write heads
- Landing Zone / Auto Parking: Resting the head on the surface when not spinning in an area where there is no data
  - → In olden times: Manual. Today fully automatic
- Impenetrable to dust, but not airtight
- „Geometry“
  - → Number of platters, heads, cylinders, sectors
- Reserve tracks to enable size guarantee (every disk has phys. errors!)
- SMART = Self-Monitoring Analysis and Reporting Technology

Tracks and sectors



- Formatting the disk creates a filesystem on the media
  - → Which must be able to address individual "parts"!
- A disk is divided into (thousands) of concentric circles = tracks
- Each track is subdivided into sectors of each 512 bytes
  - → Not every track has the same number of sectors, however!
- sector = The smallest addressable unit on a disk (="parts")
  - → Because of various reasons, larger units might be created on higher levels
    - » Example: Clusters, partitions, directories, files, …

Tracks and sectors

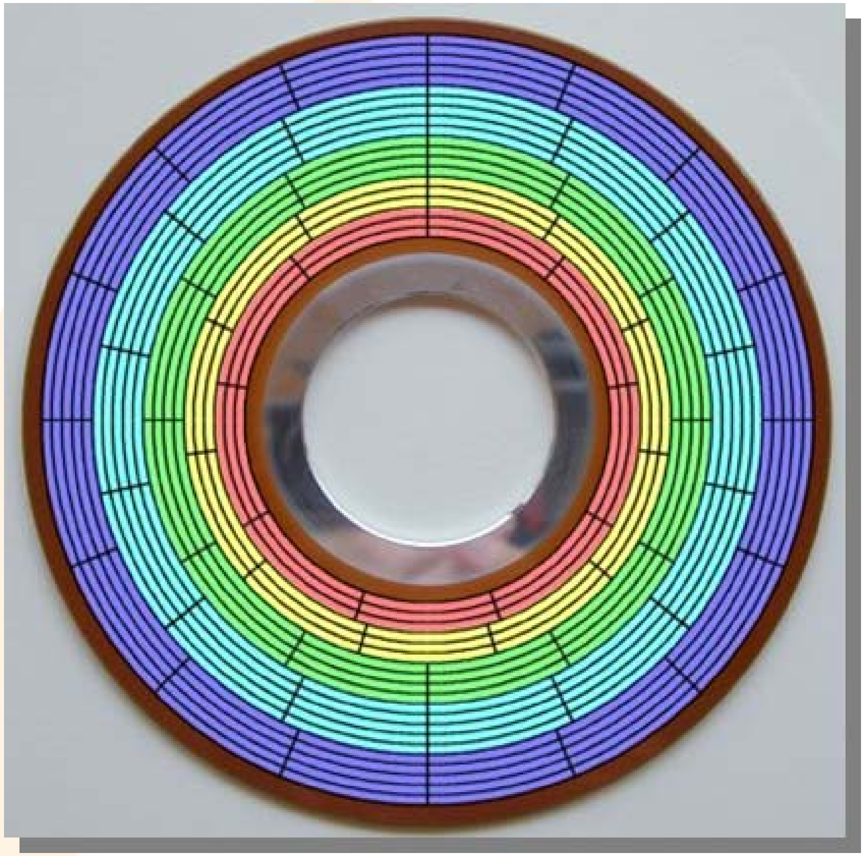


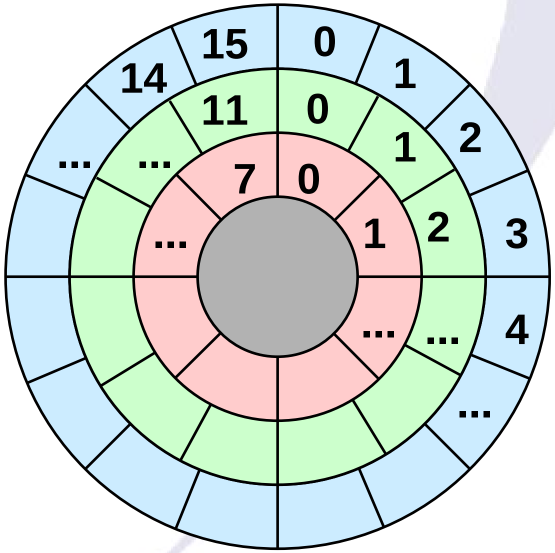
- 5,25" disk
  - → 2 sides
  - → á 40 tracks
  - → á 9 sectors
- Space for data:
  - → 2*40*9*512
  - → 368640 Bytes
    - » =360 kBytes

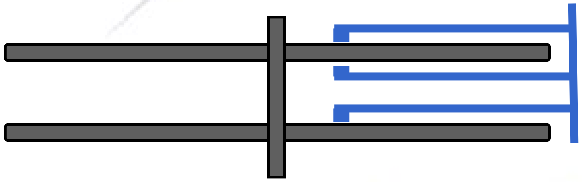
Image: 20 tracks, 16 sectors

Source: http://www.storagereview.com/guide2000/ref/hdd/geom/tracks.html

- Zones with different number of sectors per track
  - → Why not different for each track? → Because, …





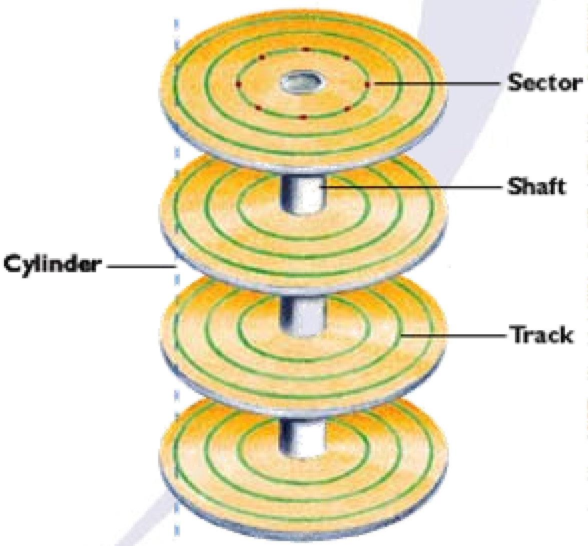


Source: http://www.storagereview.com/guide2000/hdd/...

Cylinders

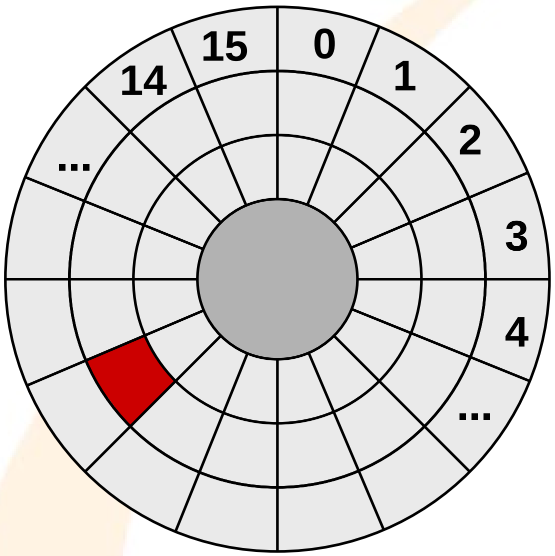
- All tracks on a harddisk which are aligned
  - → A harddisk may consist of several physical disks (=platters)
  - → All physical disks spin at the same rate and synchronously (=common shaft)
- Accessing data on the same cylinder is possible without moving the heads!
  - → All heads are mounted on a single actuator arm → Simultaneous moves
- Example: A cylinder of a harddisk with 4 platters consists of 8 tracks

Tracks, Cylinders, and Sectors



Physical structure of platters

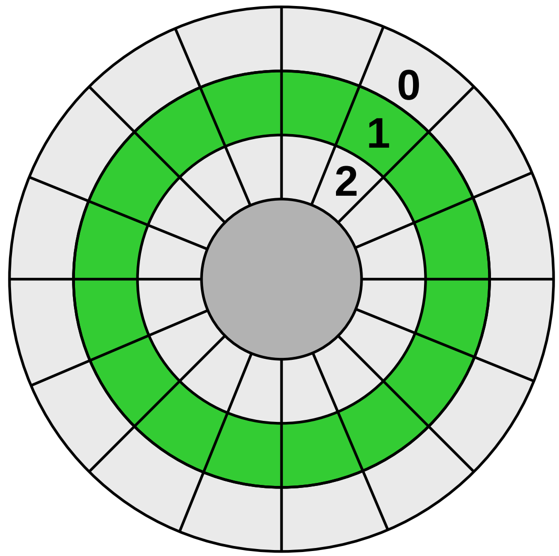
Sector



sec_per_track
(16)

sec
[0 .. sec_per_track-1]

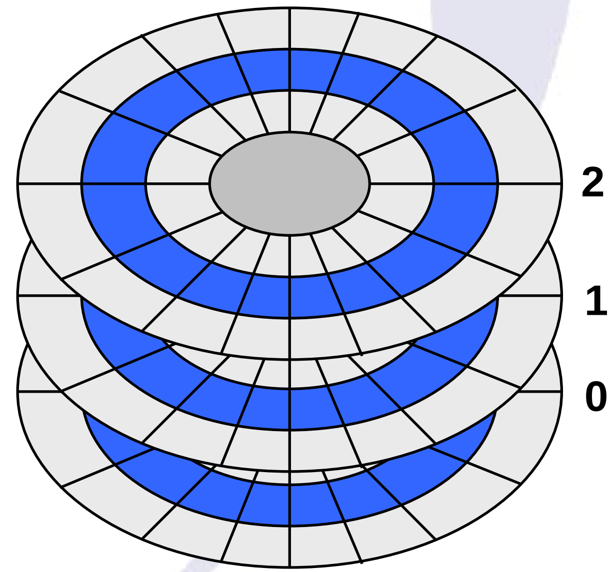
Track



nr_cyl
(3)

cyl
[0 .. nr_cyl-1]

Cylinder



tracks_per_cyl
= Number of heads
(3)

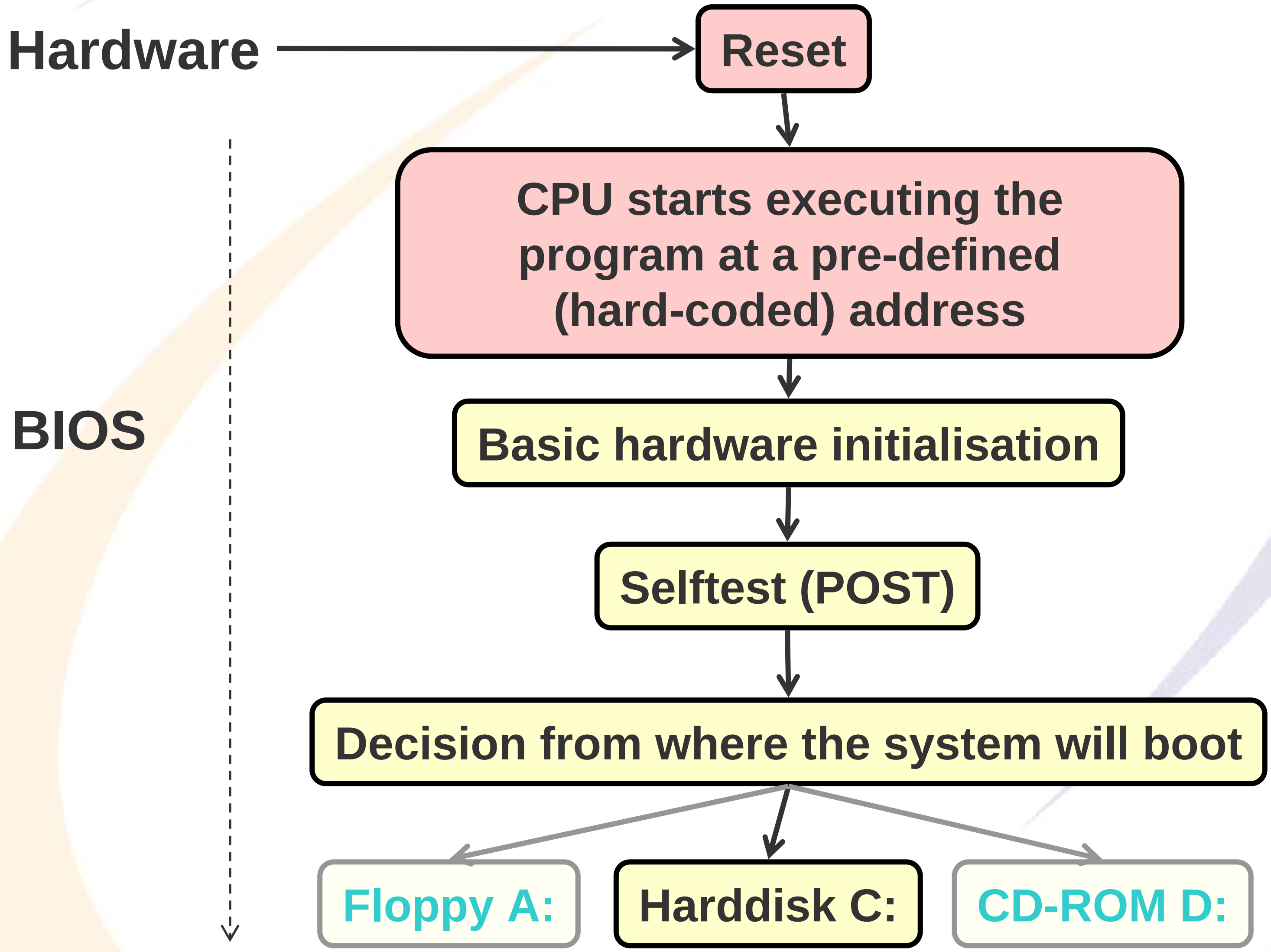
head
[0 .. tracks_per_cyl-1]

Introducing "clusters"

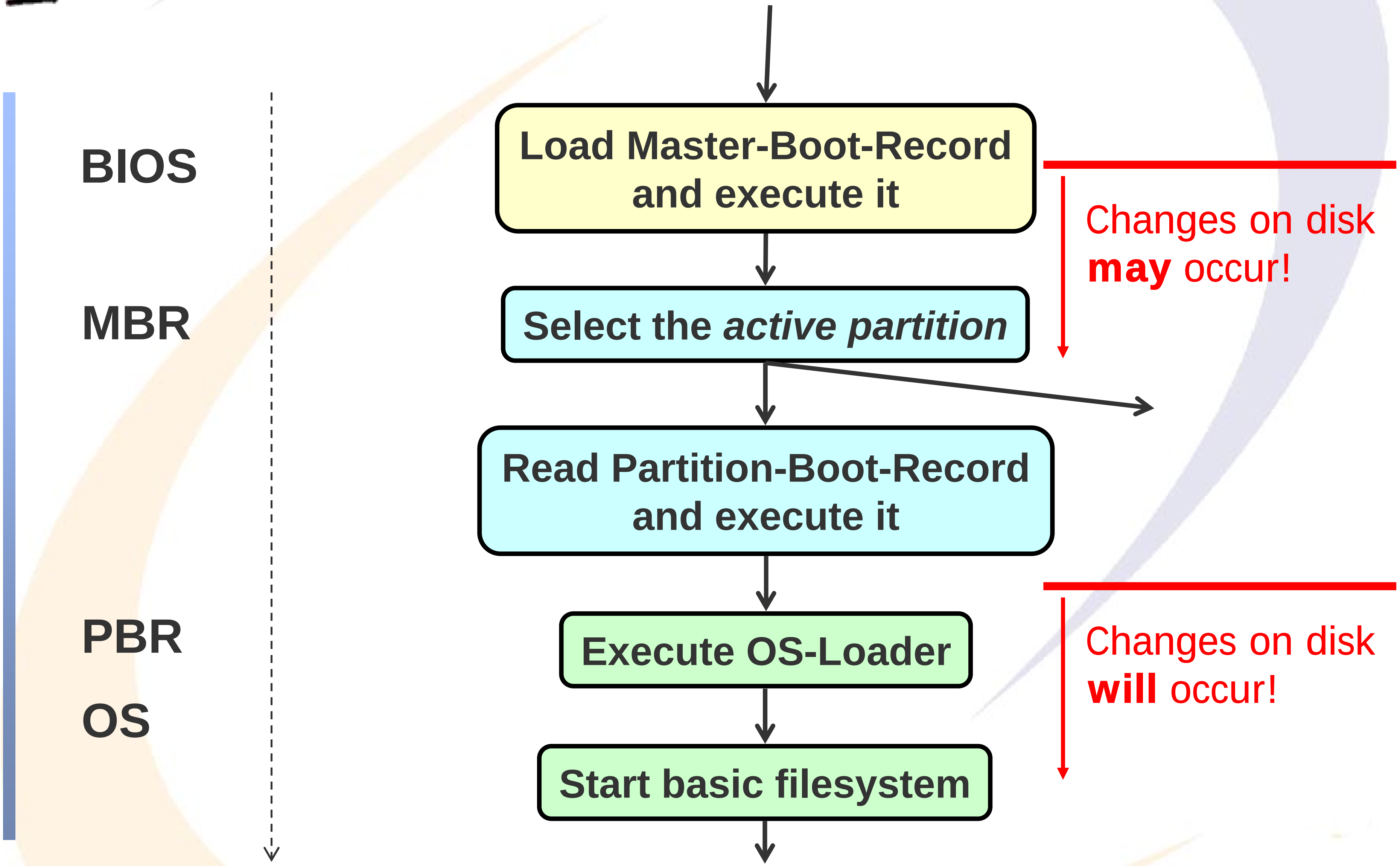
- Several sectors are combined to a single cluster
- Cluster = Smallest part which can be addresses individually by the operating system
- Introduced to manage large/variable-size harddisks by OS
  - Example: FAT16 can only address 2^{16} units
    - » 1 unit = 1 sector → 32 MB
    - » 1 unit = 1 cluster (=4 sectors each) → 128 MB
- What about fragmentation?
  - Internal fragmentation: Space between end of file and end of cluster
    - » Increases: File slack → Forensic!!!
  - External fragmentation: Clusters are not allocated in "sequence"
    - » Reduced slightly, as less "units" are needed for a single file
- Advantages and problems of cluster size?
  - A 1 byte file requires at least a full cluster
    - » Depends strongly on the number of small files!
  - Larger disks are possible

Disk-Partition and OS-BOOT

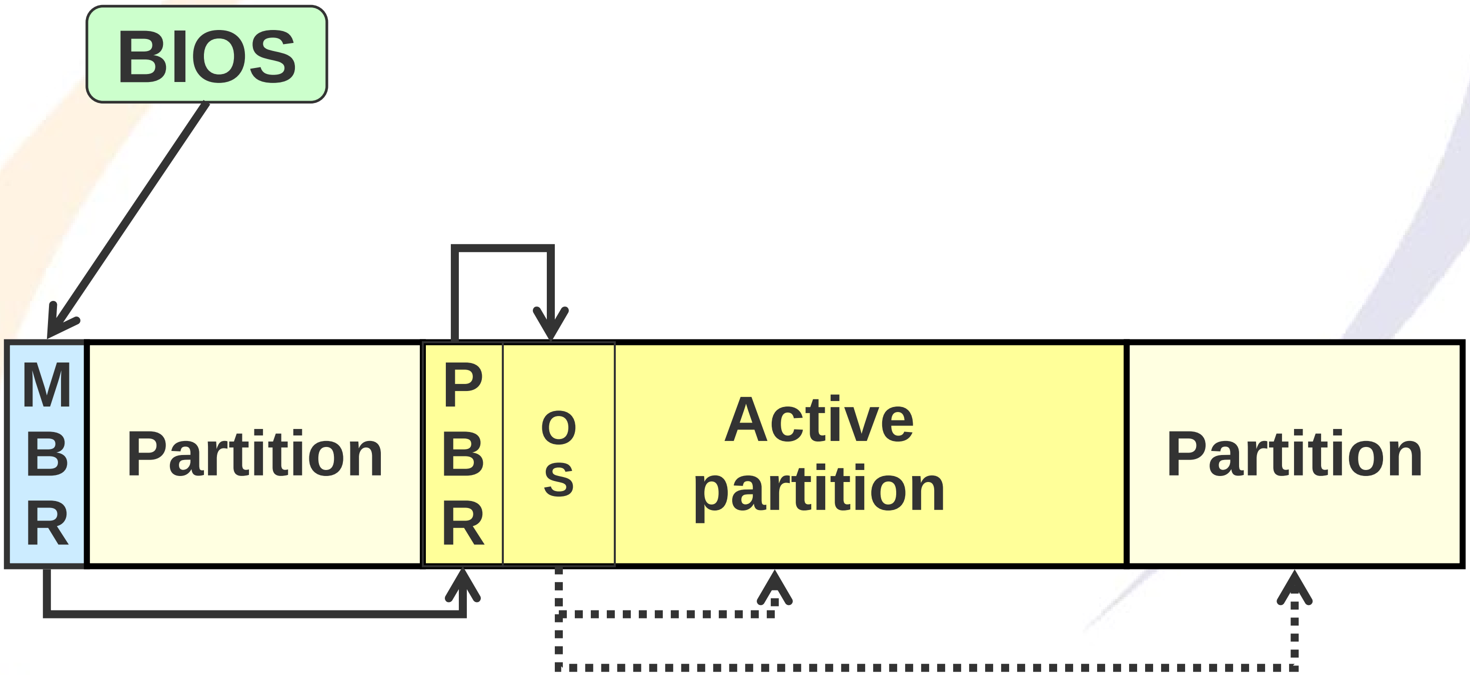
- BIOS
  - → „Basic Input / Output System"
  - → Provides also information on disks
  - → Cannot be changed by a program
    - » Modern computers: Flash-programmable, but often requires setting a jumper on the motherboard to enable this!
- MBR
  - → Master Boot Record
  - → Contains partition information on the disk and a small piece of code (initial loader for the operating system)
    - » This piece of code is executed first → Boot sector viruses!
  - → Contains the partition table
    - » List of partitions; which is active, set as boot, …
  - → Located at Cylinder 0, head 0, sector 1 (harddisks, floppy disks)


Hardware
Reset
BIOS
CPU starts executing the program at a pre-defined (hard-coded) address
Basic hardware initialisation
Selftest (POST)
Decision from where the system will boot
Floppy A:
Harddisk C:
CD-ROM D:

OS BOOT (2)


BIOS
MBR
PBR
OS
Load Master-Boot-Record and execute it
Select the *active partition*
Read Partition-Boot-Record and execute it
Execute OS-Loader
Start basic filesystem
Changes on disk **may** occur!
Changes on disk **will** occur!

- Boot sequence



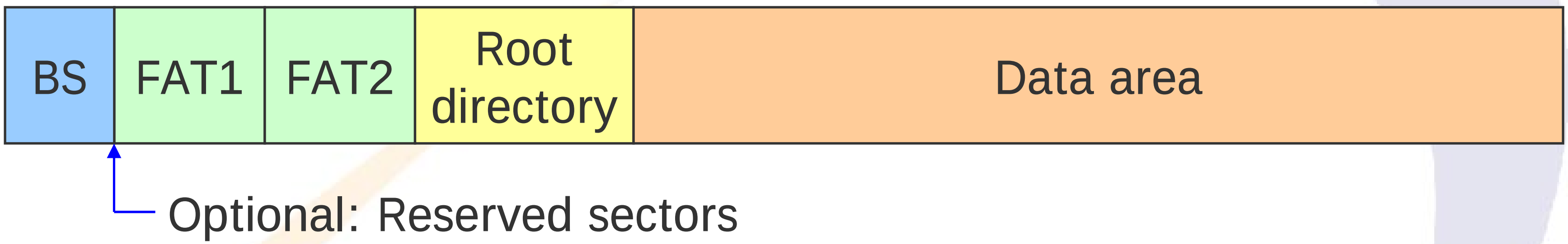
The FAT file system

- Very old: Was developed by Microsoft for MS-DOS
  - → Partially patented!
  - → Little overhead
  - → Used today still for memory sticks, flash drives, etc.
    - » Not used anymore for "main" OS partitions (NTFS, etx, …)
- Big advantage: Standardized
  - → This means, available fully on various OS!
    - » NTFS can be used on Linux, but not completely
    - » Ext can be used on Windows, but not completely
- Various versions exist: FAT12, FAT16, FAT32
  - → FAT16: Typically used on most flash disks etc.!
  - → We will only discuss FAT16 here!
- Bad sectors are marked as such only within the cluster
- Simple and fast for smaller disks!

Properties of FAT16

- Stores only short filenames: 8.3
  - → Long filenames possible through a (patented) extension
- Stores creation, modification and access date
- Attributes: Read-only, hidden, system, archive
- Maximum number of files: 65517
  - → FAT 12 → 2^{12}, FAT 16 → 2^{16}, FAT32 → 2^{28}
  - → Root directory: Typically 512 files; maximum 32767 files
    - » Fixed maximum size; created during formatting
- Maximum file size: 2 GB
- Maximum volume size: 2 GB (theoretical: 4 GB)
- Allows hierarchical directories
  - → Each counts against the limit as a file

Physical layout of FAT16

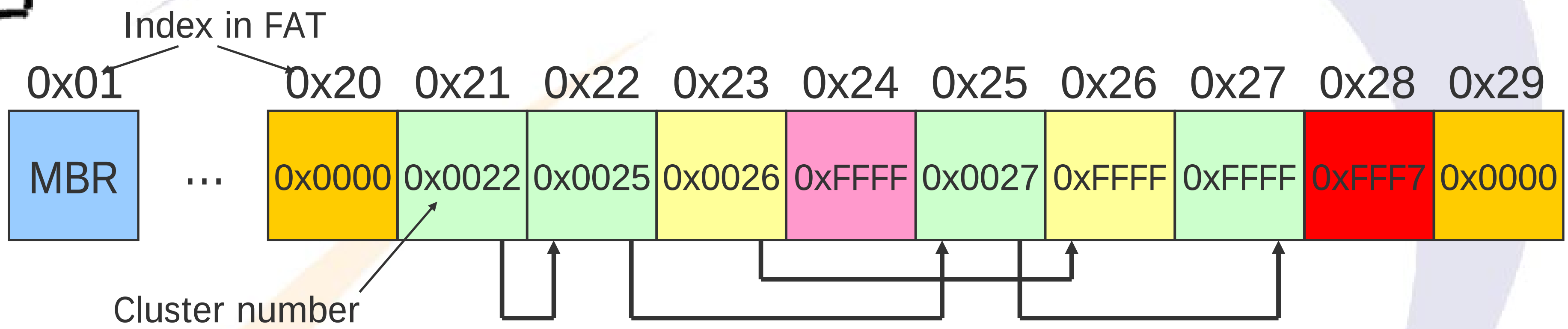


- **Boot sector: A single sector containing the boot code and the partition table**
  - → **More reserved sectors immediately afterwards possible**
- **FAT1: The File Allocation Table**
  - → **Contains the map to the data area (which clusters used)**
- **FAT2: Copy of FAT1**
- **Root directory (fixed location!)**
  - → **Location and properties of files**
    - » **Note: Subdirectories are located in the data area!**
- **Data area: Where files and subdirectories are located**

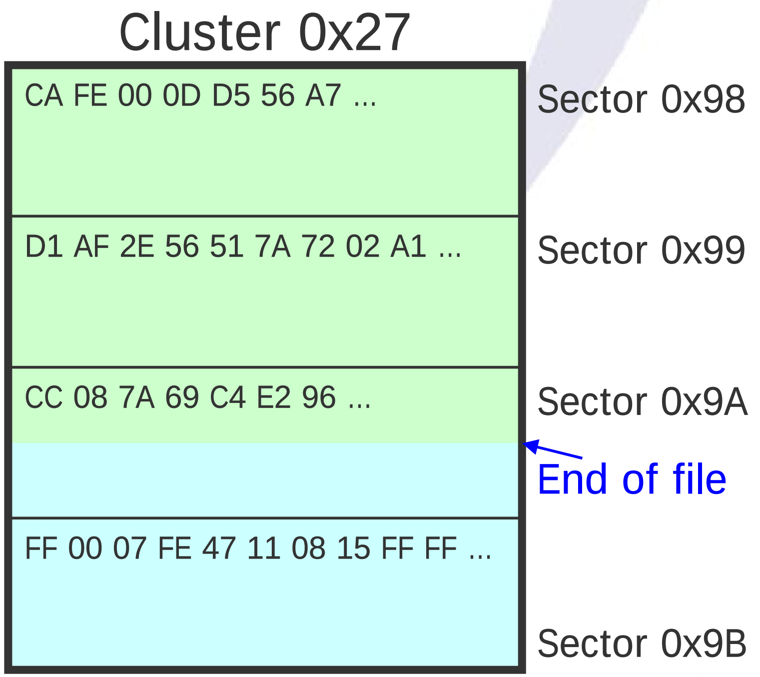
The File Allocation Table (FAT16)

- Basic concept of storing/accessing a file:
  1. Locate file description in root directory
  2. Extract from description number of first cluster
  3. Read cluster
  4. Lookup this cluster number in FAT
  5. According to value found, go to step 3 (next cluster) or terminate (last cluster)
     » Note: FAT-lookup can also be done in a single step for a whole file and cached until all data sectors were read!
- Each cluster is described by a number as
  → Unused
  → Used by a file
  → Last cluster in a file
  → Bad cluster

The File Allocation Table (FAT16)



- The example contains 3 files
- Directory entries:
  - → A: Start cluster 0x21
  - → B: Start cluster 0x23
  - → C: Start cluster 0x24
- Cluster 0x28 is erroneous
- Cluster 0x20 and 0x29 are free



Storing a directory in FAT16

- Like normal file, but format identical to root directory
  - → 11 bytes: Name (8.3)
  - → 1 byte: Attributes
  - → 5 bytes: Creation time and date
  - → 2 bytes: Last access date (no time!)
  - → 4 bytes: Last modification time and date
  - → 2 bytes: First cluster number
  - → 4 bytes: File size in bytes
  - → 3 bytes: Reserved
- Deleting files:
  - → Marked as deleted within the directory ONLY
  - → Marking is done by setting first filename byte to "E5h"
    - » The FAT is unaffected and can be used to reconstruct the content as long as the sectors are not reused!
    - » The rest of the directory entry remains until reused!

FAT 16 and computer forensic

- Typically, files are not actually deleted (see above)
  - → Unless the physical area is reused, it is recoverable
  - → Fragments of FAT chains may exist even then
    - » Partial recovery of files might be possible
- There is no "partition" slack within FAT
  - → All clusters are used; there are no partitions within
- Slack typically does exists
  - → Files are usually written only up to the end of the data
  - → File Slack:
    - » Data is retained from previous content in the remaining sectors of the cluster; these are not written to
  - → RAM slack:
    - » Data in the last sector of the file after its end will usually be random data from in-memory buffer; written to disk

The NTFS filesystem

- Internals are trade secrets of its creator Microsoft
  - → But commercial licensing is possible
- There are no predefined attributes for files
  - → Everything is stored as "Metadata", including filename, creation date, access permissions, …
  - → This allows easy extension to other associated data
- Names are stored as 16 Bit/Character → UTF-16 possible
  - → But not restricted to it, any 16-Bit values are allowed
- Organisation is in a B-Tree
  - → Allows very fast searching for huge numbers of elements
    - » Drawback: Complex to implement
- Journaling is built-in
  - → However, only for the filesystem itself, not the data
    - » The directory will be correct, but the file may be garbled!

Properties of NTFS

- Some file names are not allowed
  - → Reserved for internal management; all start with "$"
    - » Examples: $MFT, $MFTMirr (Master File Table & its mirror)
- Maximum volume size:
  - → 2^{32}-1 clusters (implemented); 2^{64}-1 clusters (theoretical)
  - → With 4 kB cluster size → 16 TB
  - → Note: The boot partition is typically limited to 4 GB as it is initially FAT (and converted to NTFS later)!
- Maximum file size:
  - → ≈ 16 TB (implemented); ≈ 16 EB (2^{64}-2^{10} B; theoretical)
- Compared to FAT there is no date restriction
  - → Range from 1.1.1601 – 28.5.60056
- Suffers from defragmentation problems
  - → The defragmentation API only allows relocating 16 clusters at once and only every 16 clusters of a file

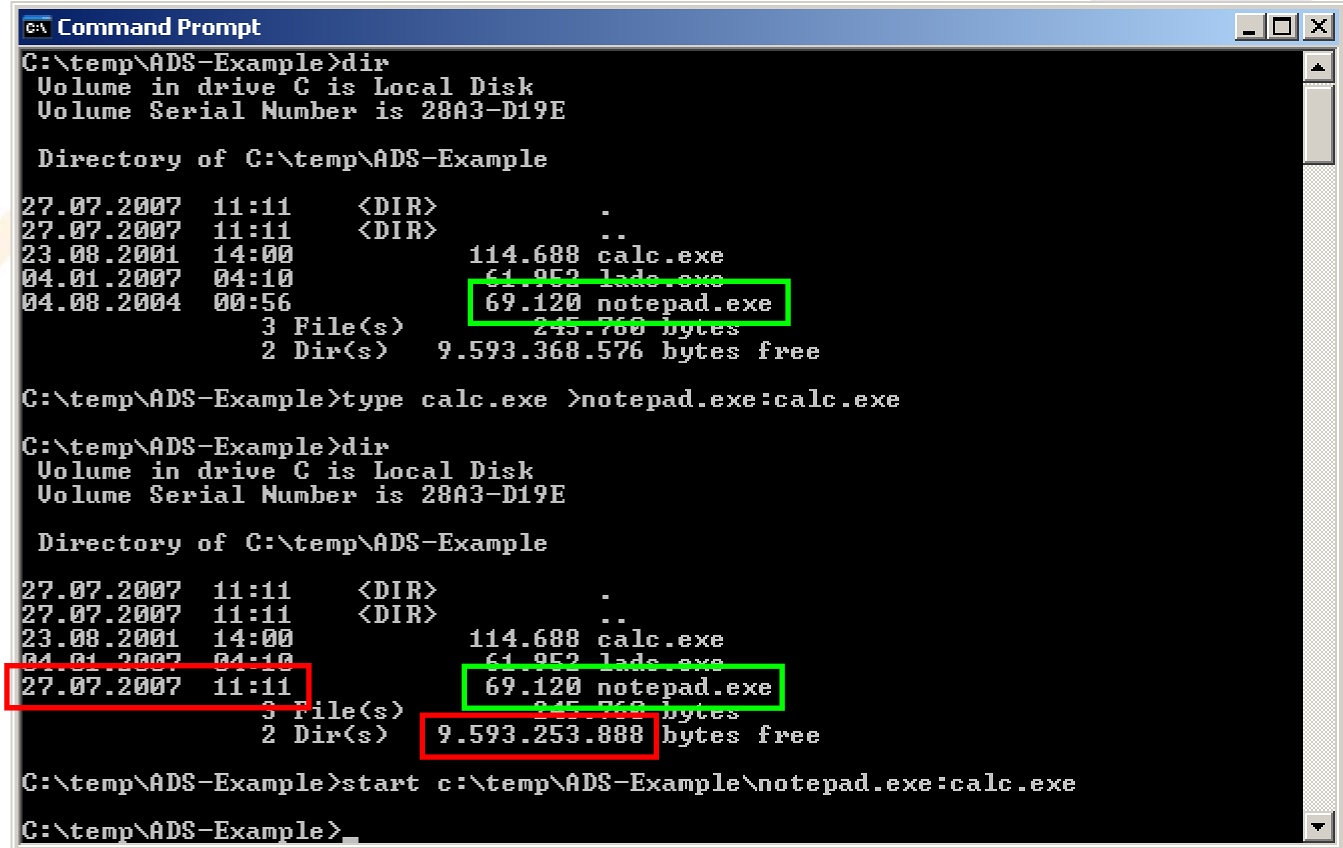
Master File Table (MFT)

- Contains the "directory" structure and the files
  - → Located at the beginning of the disk in a reserved space
  - → If it grows too much, it is extended to the data area
- Contains file records of fixed size
  - → These are reused after deletion
  - → A reserved area for system files exists
- File records:
  - → Each file has at least one with the "standard" attributes
  - → More space needed? → More records allocated to file
  - → Contains e.g. information on access rights
- Updates are first logged, then performed, then marked as completed in the log → Journaling

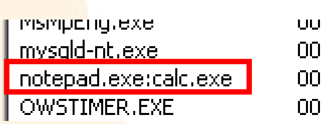
Alternate Date Streams (ADS)

- Additional "attributes" of a file: This can be a file itself!
- Attention: In the "normal" UI these are invisible!
  - → The file shows up identically in the GUI
  - → The file shows up identically on the command line
    - » Note: The file size stays the same!
  - → The file behaves exactly as it did before
  - → They show only up in the taskmanager in recent versions
  - → What changes is the modification timestamp
- Alternate Data Streams cannot be disabled or limited
  - → Only "normal" access restrictions of the base file apply
  - → But copying the base file to a system without ADS will automatically strip them

ADS example



Taskmanager:



NTFS security

- NTFS contains access permissions
  - → Without the correct permission, no access is possible
    - » Use direct (hex) access to the disk
  - → Alternative: Insert (copy of) disk into system where you are the administrator
    - » Reason: The administrator can reset permissions!
      - – These are then lost (→ copy!), but you get access to the file
- NTFS support file encryption
  - → Specifically targeted at making the disk "unreadable" by third persons (typically thieves, but includes CF!)
  - → Files are encrypted separately, i.e. only their content
  - → The key is stored for each user and with recovery agents
    - » Typically the administrator
    - » Newer version require admin rights and the users password!
  - → Tools can decrypt, but >= XP SP1 the recovery agent's password is needed

NTFS and computer forensic

- General consideration like File-/RAM-slack apply as well
- NTFS supports "Volume Shadow Copies"!
  - → Intended for backups of open files
  - → Keeps "old" versions of files
  - → When the file is written to, the previous values are copied to another place; on reading it is "overlaid" back
  - → These shadow copies reside on the disk and can therefore contain copies of older version/deleted files!
- Special software needed for interpretation
  - → As no specification is freely available and the structure is complex in itself
- Bitlocker (Vista) may require live gathering!
  - → May be configured so it asks for password before boot!
    - » Whole disk is encrypted, i.e. no NTFS structures readable

The EXT3 filesystem

- EXT3 is EXT2 + enhancements
  - → This means, the EXT2 tools also work on EXT3!
  - → Added:
    - » Journal: For crash-resistance
    - » Tree-based directory indices: For very large directories
    - » Online filesystem growth: Enlarging "on the fly"
- EXT3 is based on "inodes" (and blocks=clusters)
  - → Contains metadata (file size, dates, …)
    - » But not: Filename (→ in directory)!
  - → Links to the actual data blocks
    - » These may be direct or (1-N) levels of indirection
      - – Indirection: Pointer to block containing pointers to data blocks
      - – EXT3: 12 direct, 1 single indirect, 1 double ind., 1 triple ind.
  - → Reference counter (for links)

Properties of EXT3

- Maximum volume size: 16 TB (4 kB block size)
- Maximum file size: 2 TB (4 kB block size)
- Maximum filename size: 255 Bytes
  - → May contain all characters except 0x00 and '/'
- Stores modification, attribute mod., and access time
- No real defragmentation or online compression
- An EXT3 partition is subdivided into block groups
  - → Block count per block group is variable
  - → Determined on formatting
- "Clusters" are called "blocks" in EXT3
  - → The block size is determined on formatting: Typ. 4 kB

EXT3 physical layout

Partition:

Boot sector	Block group 1	Block group 2	Block group 3	Block group 4	Block group 5	…	Block group N

Single block group:

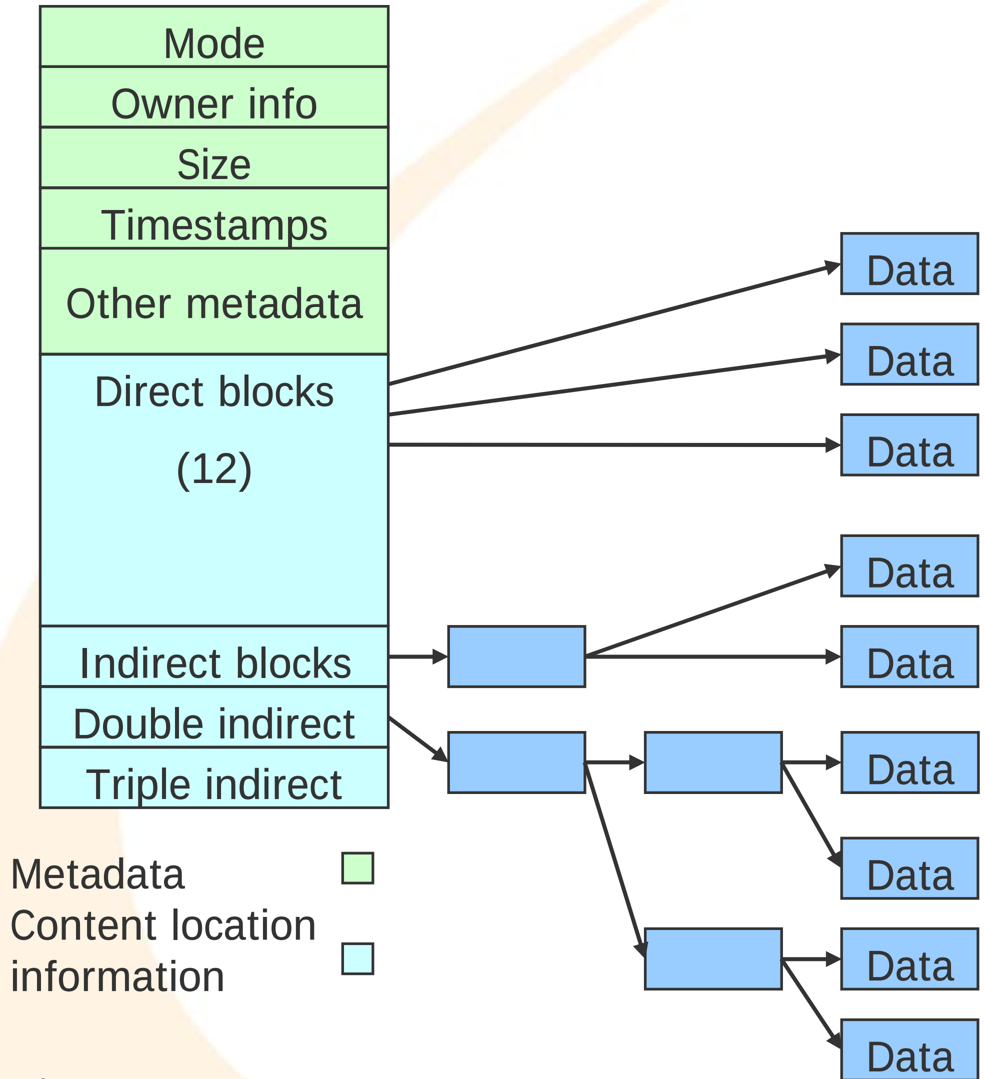
Super block	Group descriptors	Block bitmap	Inode bitmap	Inode table	Data block 1	Data block 2	Data block 3	Data block 4	…	Data block N

- **Each block group contains redundant copy of general information structures (superblock + FS descriptor)**
  - → **Block+Inode bitmap, Inode table: Only for this block group!**
  - → **Block groups reduce the distance between file information and file data**
    - » **This is not a hard allocation: Data from a file can also be in a different block group!**
  - → **"Sparse superblocks": Repeated only in some groups to reduce space used on large volumes**

Block and Inode bitmaps

- Block bitmap: Which blocks are used/free
  - → Every block is represented by a single bit (→ bitmap)
  - → Organization:
    - » 1 = used, 0 = free
    - » Block 1 = Byte 0 Bit 0, Block 2 = Byte 0 Bit 1, Block 8 = Byte 0 Bit 7, Block 9 = Byte 1 Bit 0
- Inode bitmap:
  - → Every Inode is represented by a single bit
  - → Organization: Like block bitmap
    - » The first bits are always set: Superblock, group desc., …!

Inodes



- Mode: Permissions
  - → Includes Inode type
    - » File/Directory/Link/…
- Owner info:
  - → User and group ID
- Size: File size in Bytes
- Timestamps:
  - → Access time
  - → Creation time
  - → Modification time
  - → Deletion time
- Other metadata:
  - → Link/Block count
  - → File flags
  - → …

EXT3 Undelete

- EXT3 undelete is very difficult
  - → File size and block addresses are overwritten on delete!
    - » Reason: Easier recreation through journal after crash
    - » Result: File name still exists, file data still exists, but which blocks of data belong to the file in which order is lost
  - → Undelete is still possible, but it must work on the level of individual blocks/clusters, not just "unmarking the directory entry as deleted"!
    - » Basis: Journal entries or "file carving"!
      - – Journal: Several inodes/block; Whole block is saved in journal → Journal entries for other files may contain the pointers!
      - – Carving: Try to detect start/end of file by "magic numbers"
    - » Note: These approaches identify only parts of the file. The rest must be assumed to be "physically in between"!
      - – This fails when the file is fragmented → Undelete very difficult!

EXT3 directory

- Directories are "ordinary" files
  - → Root directory: Inode number is part of superblock!
  - → They contain no metadata at all → Inode
- Format is very simple:
  - → Inode associated with file (4 Bytes)
  - → Length of this entry in bytes (2 Bytes)
  - → Filename length in bytes (1 Byte)
  - → File type (1 = file, 2 = directory, 7 = Symlink, …; 1 Byte)
  - → Filename (N Bytes)

0	4	6	7	8	9	13	15	16	17
0x0815	0x09	0x01	0x02	.	0x4711	0x0A	0x02	0x02	..

19	23	25	26	27
0xCA78	0x10	0x08	0x01	User.txt

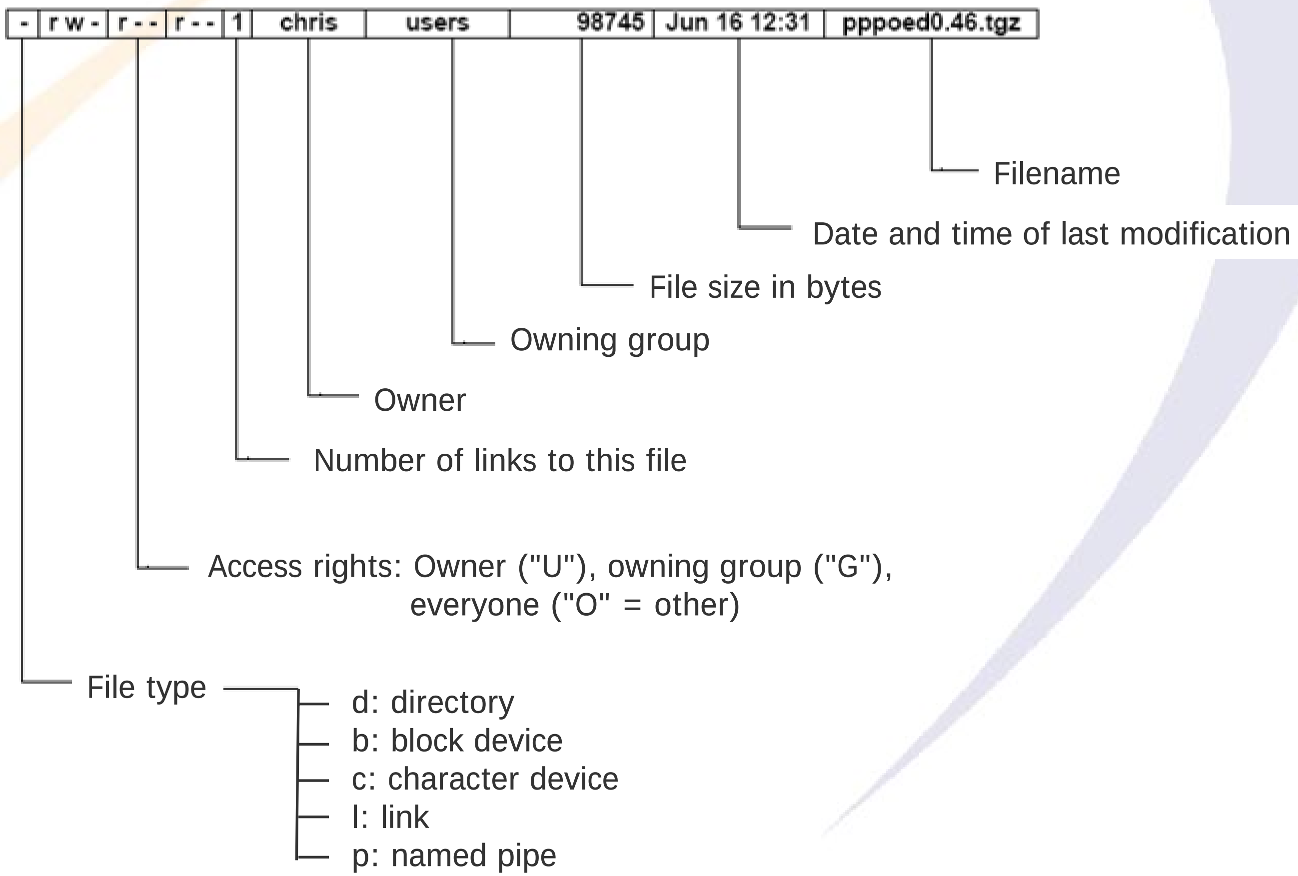
Note: Each record is usually aligned to 4 Bytes. This is not shown in the image!

EXT3 security

- The traditional unix rights system:
  - → There are users and groups
  - → Each user is member of a single primary and an arbitrary number of secondary groups
  - → One special user („root“), has all rights on (normal) files or can obtain them through changing ownership/rights
  - → Each file has an owner and an "owning group"
  - → There are only 3 permissions: "read", "write", and "execute"
  - → A combination of these three permissions can be set for three different groups of persons:
    The owner, the owning group, and for everyone
  - → Additionally there are a few specialty bits
    - » E.g. executing the program as owner/owning group, regardless of the actual user

EXT3 security example

Command: **`ls -al`**



Access control lists

- ACLs also exist, but on a different layer
  - → Supported by: Ext2, Ext3, XFS, JFS, ReiserFS
- The normal permissions (rwx) of a file can be assigned to arbitrary other users and groups
  - → Commands: getfacl, setfacl
- Example:
  - → "getfacl index.html"
  - → # file: index.html
    # owner: root
    # group: apache
    user::rw-
    user:sonntag:rwx
    group::r--
    other::---

Attention: Filesystem must be mounted accordingly for this to be supported (/etc/fstab !)

EXT3 and computer forensics

- EXT3 is a journaling filesystem
  - → Depending on the mode used, file metadata and perhaps even file data may be present in the Journal!
    - » This is actually a problem for wiping too …
  - → Making a copy of a live system is difficult
    - » Special tools needed or remounting as read-only!
- Recovering deleted files can be very difficult
- General consideration like File-/RAM-slack apply as well
  - → But swap space is a separate partition, not a file, and therefore itself a "filesystem"

Conclusions

- Recreating evidence from a filesystem requires intimate knowledge of the filesystem or special tools
  - → An important approach is "file carving", i.e. recreating files through assembling only data sectors and ignoring all directory entries
    - » This is much more independent of the file system, but also more difficult; e.g. which sectors belong to a binary file
      - – Plain text files → Easy!
  - → Many different filesystems exist, but only few are common
    - » "Rare" filesystems might pose special difficulties!
- Journaling file systems offer an additional approach
  - → Some data might be present in the journal
    - » E.g. recently deleted data

Questions?

Thank you for your attention!

Literature

- Alternate data stream http://www.wikistc.org/wiki/Alternate_data_streams
- Berghel, H., Brajkovska: Wading into Alternate Data Streams. Communications of the ACM Aplril 2004/Vol. 47, No. 4, 21-27 http://portal.acm.org/ft_gateway.cfm?id=975836&type=pdf&coll=GUIDE&dl=ACM