Persistent Storage

- Datastructures and Algorithms
L 05: Case Study: FAT Filesystem
Questions

• Why is consistency important?
  • Types of machine failures
  • Consequences?

• What are the ACID definitions?
  • A?
  • C?
  • I?
  • D?

• The 3 classes of consistency mechanisms?
Questions (2)

• Types
  • Second Copy
  • Log
  • Copy-on-Write

• Questions for each type
  • What is necessary to make it work?
  • Storing of a new set of informations, how?
  • Recovery at reboot, how?
FAT: History

• Predecessor: Data in CP/M on Floppy Discs
  • No subdirectories.
  • Defined access function for 8 inch discs (fixed)

• Development:
  • CP/M -> QDOS
  • QDOS -> 86-DOS (also CP/M 86)
  • Microsoft clone of for IBM PCs: PC-DOS
  • Renamed to MS-DOS when sold independent of IBM

• The only relevant news in DOS was FAT
FAT: History (2)

• CP/M still used FCBs
  • 128-byte records on 8 inch floppies
  • FCBs still were there in MS-DOS 1

• MS-DOS 2:
  • File handles (like in other OSes like Unix, ...)
  • Sub-directories
FAT: History (3)

- CP/M access to disk via BIOS
  - in MS-DOS: IO.SYS
- CP/M: BDOS (Basic Disk Operating System)
  - in MS-DOS: MSDOS.SYS
- CP/Ms command line interface: CCP (Console Command Processor)
  - MS-DOS: COMMAND.COM
- CP/M: Porting of software was easy
  - Only BIOS calls were necessary for IO
  - Pointers to BIOS calls on a defined address in memory (real memory, no virtual memory support)
FAT Filesystem: History (4)

- FAT12: 1976-77 (QDOS, without subdirectories, they were introduced in MS-DOS 2.0)
- FAT16: September 1987 (Compaq DOS 3.31)
- FAT32: August 1996 (Windows 95b, also called OSR2)
FAT Filesystem: Disk Structure

- Structure on the media/disk (in sectors):
  - Boot sector
  - (Reserved Sectors)
  - FAT
  - Root directory
  - Data
FAT Filesystem on one page

• Structuring elements:
  • Root sector:
    - Structure information about the filesystem
  • Directory contents
    - Table (like dataset structure FB)
  • File allocation
    - Linked List
  • Bad blocks, free blocks
    - Cluster tagging
FAT Filesystem: Bootsector

- Defines the size of the used space
- Defines the cluster size
  - All other parts are made of clusters ($2^x \times$ sectors)
    - Directory, FAT, data
- Position of the other parts
  - FAT, Directory, data
  - Implicit: how many data clusters are available

- For more, check Wikipedia:
  http://en.wikipedia.org/wiki/File_Allocation_Table
FAT Filesystem: Sizes

- **FAT12 (floppy disks):**
  - Cluster size: 512 – 4096 Byte
  - Max. disk size: 32MB

- **FAT16 (disks, today: mobile disks, ):**
  - Cluster size: 512 Byte – 32 KByte
  - (NT and Enhanced DR-DOS also used 64 KByte)
  - Max. disk size: 2 GB (NT, DR-DOS: 4GB)

- **FAT32 (disks, today: larger mobile disks)**
  - Cluster size: 512 Byte - 32 Kbyte
  - Max. disk size: 2 TB – 16 TB (64 KB Cluster)
  - Max. file size: 4 GB
FAT Filesystem: FAT

FAT = File Allocation Table

- The table contains entries of different types
  - The number of next cluster (like a linked list)
  - End of the cluster linked list
  - Defect cluster
  - Reserved cluster
  - Free cluster

- The table exists 2-times
  - Sometimes used to help diskchk
# FAT values

<table>
<thead>
<tr>
<th>Type</th>
<th>FAT12</th>
<th>FAT16</th>
<th>FAT32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>0x000</td>
<td>0x0000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x001</td>
<td>0x0001</td>
<td>0x00000001</td>
</tr>
<tr>
<td>Next Cluster</td>
<td>0x002-0xFEFE</td>
<td>0x0002-0xFFFF</td>
<td>0x00000002-0x0FFFFFF</td>
</tr>
<tr>
<td>Defect</td>
<td>0xFF7</td>
<td>0xFFFF</td>
<td>0xFFFFFFFF</td>
</tr>
<tr>
<td>End</td>
<td>0xFF8-0xFFF</td>
<td>0xFFF8-0xFFFF</td>
<td>0x0FFFFFFF8-0x0FFFFFFF</td>
</tr>
</tbody>
</table>
FAT Filesystem: Root-Directory

• Number of clusters after FATs
  • FAT12 and FAT16, root directory has a fixed size
  • FAT32: anywhere in the data clusters
    1st cluster of root directory noted in root sector

• Every entry has a size of 32 Byte
  • The clusters of the root directory are filled
    - like with the FB datasets
**Root-Directory: Entry**

- **Name** (8 Byte)
- **Extension** (3 Byte)
- **Attribute** (1 Byte)
- **Reserved** (1 Byte)
- **Date and time of creation** (5 Byte)
- **Date of last access** (2 Byte)
- **First Cluster (upper 2 bytes)** (2 Bytes)
- **Date of Last Change** (4 Bytes)
- **First Cluster (lower 2 bytes)** (2 Bytes)
- **Size** (defines the limit) (4 Bytes)
Root-Directory: Name Prerequisites

• Naming conventions:
  • A-Z, 0-9, space, !, #, $, %, &, ', (, ), -, @, ^, _, `, {, }, ~
  • Plus values 128-255 (upper ASCII halve)
• First Character has special rules:
  • 0x00: Empty entry
  • 0x05: Kanji
  • 0x2E: . or ..
  • 0xE5: File had been deleted
    - Can be used for new files
    - Clusters should be free
Root-Directoy: Attributes

• An Or-ing of
  • 0x01: ReadOnly
  • 0x02: Hidden
  • 0x04: System
  • 0x08: Volume Label
  • 0x10: Subdirectory
  • 0x20: Archive
  • 0x40: Device (internal use only)
  • 0x80: Unused
  • 0x0F: Used in Long-File-Names
Root-Directory: Long Names

• Using a “trick”:
  • One or multiple (up to 20) long name entries
    - Before the file entry itself
  • Each long name entry contains up to 13 UTF-16 Chars
    - Each UTF-16 character needs 2 Bytes
    - All long name entries are concatenated

• Long name entry:
  - Attribute is set to 0x0F (ro + hidden + system + volume)
  - Cluster start is always 0
  - Last Char is always a 0x0000

• The file has also a unique short name (8+3)
**Root-Directory: Long Name Entry**

- These entries therefore look like:
  - 0x00: Sequence number
  - 0x01: 5 UTF-16 Chars
  - 0x0b: Attribute: 0x0F
  - 0x0c: always 0x00
  - 0x0d: Checksum of DOS file name
  - 0x0e: 6 UTF-16 Chars
  - 0x1a: always 0x0000
  - 0x1c: 2 UTF-16 Chars
Root-Directory: Long Name Example

- Sequence number:
  - The “wrong” entries are written in reverse order in front of the real DOS entry, the biggest has 0x40 "or"ed to the sequence number

- Example:
  - 0x43: “me.ext”
  - 0x02: “y long filena”
  - 0x01: “File with ver”
  - .......: FILEWI~1.EXT (or some such)
Root-Directory: Entries

- For more, check Wikipedia:
  http://en.wikipedia.org/wiki/File_Allocation_Table
Fat Filesystem: Data

- First Cluster number is noted in directory
- Sector position of cluster can be computed
  - Cluster size from root sector and cluster number
- Next Cluster:
  - Read the entry in FAT which corresponds to this cluster
FAT Filesystem: Subdirectory

- Subdirectories are stored like ordinary files
  - Attribute 0x10 is set
- Same structure as root directory
Patents

- Microsoft owns 4 patents on long file names in FAT32. But these have been partially reverted.
- Still: There has been an off-court agreement between Microsoft and TomTom, where Microsoft gained access to patents owned by TomTom, and TomTom stopped usage of FAT32 (TomTom uses Linux).
Summary: Access

- Read structure from root sector
- Find the name in the directory
- Read the start cluster address from the directory entry
  - Perhaps: create the list of following clusters for fast access
- Use only the length indicated in the directory entry
Summary: FAT12

- 12-Bit Cluster numbers (max: 4077)
- Cluster size 512, 1024, 2048 or 4069 Byte
- Max. size: 32 MB
- Max. file size: 4 GB (theoretically)
- Root-Directory has 14 Clusters, and therefore max 224 entries
- Knows attributes: ReadOnly, Hidden, System, Archive
Summary: FAT16

- 16-Bit Cluster numbers (max: 65517)
- Cluster size 512 Byte – 32 (64) KByte
- Max. size: 2 (4) GB
- Max. file size: 4 GB
- Root-Directory has max. 512 entries, therefore max. number of entries in total: 65536 (one entry per Cluster)
Summary: FAT32

- 32-Bit Cluster numbers (only 28 used) (max: 268435437)
- Cluster size 512 Byte – 32 (64) KByte
- Max. size: 2 (-16) TB
- Max. file size: 4 GB
- Max. number of files: $2^{28}$ (due to number of Clusters)
- Root-Directory can be placed at random and of random size