Managing and storing digital AV assets - Dos, Donts and Dunnos

Peter Bubestinger-Steindl

`PB @ AV-RD.com`

November 22nd, 2021

Storage

Considerations

- Storage market focus may deviate from your (=preservation) use case.
- Business “longterm”: 3-5 years (=product lifecycle)
- Cross vendor compatibility? Standards? Documentation?
- **Spare parts?**
- This market focus affects prices, features and reusability.

Speaker notes

- Business focus is usually on high availability, speed and least downtime = fast replacement, not necessarily compatibility or long lasting components.

    - Spare parts: Professional hardware is good, but: Often incompatible with standard components, due to proprietary plugs, pinouts, form-factor, etc.

      I’ve seen cases where whole rack-mounted hardware servers/JBODs had to be replaced because it was not possible to get spare power supply units (PSUs) or even ventilation fans for this proprietary layout.

      If you can, get good quality hardware with vendor-neutral replacement possibilities. Available beyond 3-5 years after purchase.

"Onions have layers, Ogres have layers. You get it? We both have layers!"

Data storage has layers to consider…

Speaker notes

When considering a storage for long-term preservation, at least the following layers/components should be considered:

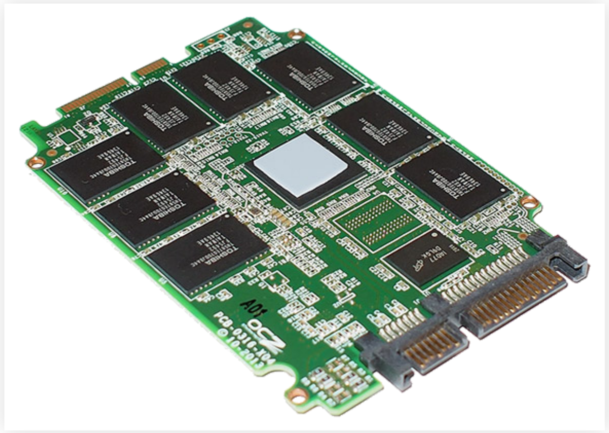
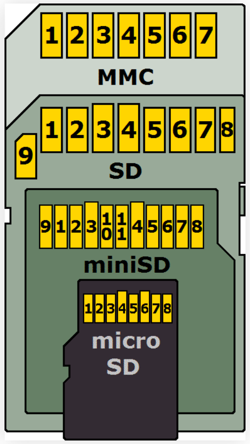
- Application required to read/write data
- Operating system
- Hardware it runs on
- Filesystem
- Drive (e.g. for tapes, optical media)
- Physical carrier (medium)

On top of that (but independent of the storage type chosen) are additional layers of the file itself:

- container format
- video format
- audio format
- metadata format
- …

Similar principles for selecting a file format apply to selecting a long-term storage solution.

Physical carrier

Speaker notes

Different carrier media have different properties when it comes to error resilience, longevity - and of course: price.

- Magnetic: Susceptible to electro-magnetic issues.
- Optical: light, temperature, warping
- Electronics: Failing capacitors
- Mechanics: Wear and tear
- Supported/obsolete interfaces or drives?

It’s usually good to mix storage media to distribute their pros/cons properties.

I won’t go into great detail here, as this is possibly covered in many other preservation-storage presentations.

This here is for the interested reader.

Examples for different use cases:

- preservation / backup:
  tape (affordable size, good preservation properties, less need for super-fast access)

- access: No need for long-term endurance, so e.g. optical would be sufficient.

- production: HDD-based storage for nearline files - tape for offline storage. Or: tape-robot with HDD cache (same concept, but fully automated)

More properties:

- Data Access Speed?
  SSD > HDD > Tape > Optical

- Cost/size ratio?
  SSD > HDD > Tape > Optical

- Widespread?
    - SSD, HDD: Excellent!
    - Tape: almost exclusive professional use case Drives not that “easy” to get (for particular LTO generation)
    - Optical: Declining (except for “cold storage” in large data centers)

In greater detail:

Optical Disks

Pros:

- Cost/size ratio: Relatively cheap (cheapest still?)
- Not susceptible to electro magnetic (dis)charge
- No mechanics/electronics in carrier - only in the drive

Cons:

- Error prone
- Suffers easily from material degradation (scratches, data layer falling off, light exposure, temperature, etc…)
- Becoming less and less common = less support, less choice, etc.

HDD: Hard Disk Drive

Pros:

- Reasonable cost/size ratio
- (yet) cheaper than SSD
- well known, well supported

Cons:

- Mechanical wear (moving parts)
- Susceptible to electro magnetic discharge
- Electronics on carrier

Data Tape

Pros:

- Cost/size ratio: Tape cartridge “cheaper” than HDD
- Tape preserves well (and there is much experience with dealing with tape degradation/aging from analogue)
- Supported in “tape libraries” (aka “tape robots”) for removing “tape-jockey” limitations.
- Larger capacity per cartridge than per HDD (yet)

Cons:

- Not common outside certain domains (large institutions, broadcast preservation, …)
- Therefore support for tapes very “specialized”
- Drives quite expensive (>3000 EUR)

Flash Memory

Pros:

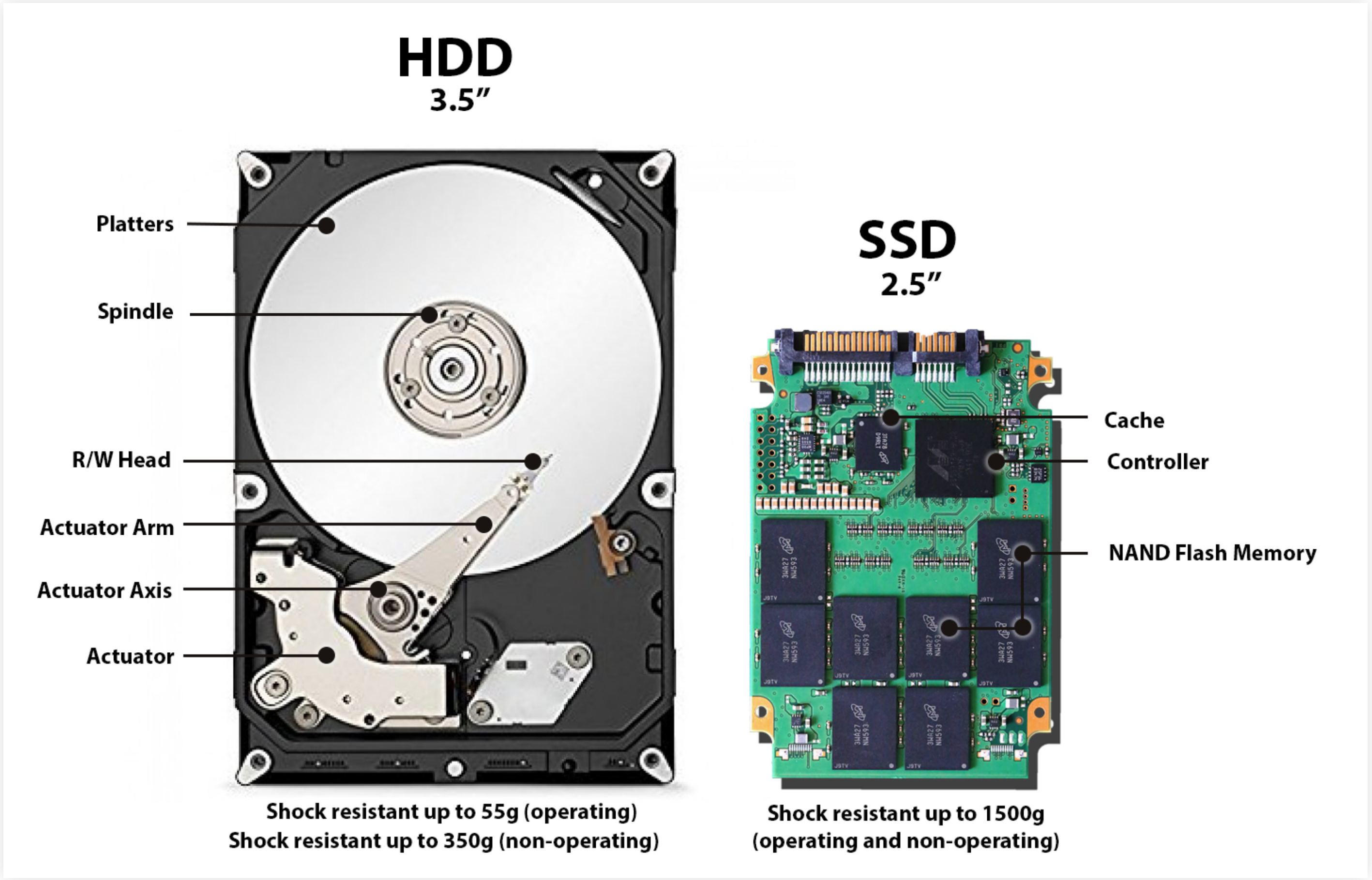
- cost/size ratio: getting cheaper, but still more expensive than HDD
- No moving parts = no mechanical wear
- SD cards: almost no electronics in the carrier
- Speed: SD cards are not sooo fast, whereas SSDs (or M.2) are extremely fast (compared to HDD).

Cons:

- Currently most expensive (still)
- Lack of long-term experience
- Limited number of write/erase cycles per flash block (becoming less of an issue) =Should not be used for large number of write operations. But preservation storage case is mostly “read”.

More about Flash memory on Wikipedia. SD card image by Steve Meirowsky (Wikipedia) - Own work, CC BY-SA 3.0

HDD vs SSD



Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)

Shock resistant up to 1500g
(operating and non-operating)

SSDs: less stuff to break

Speaker notes

HDD:

- (yet still) cheaper cost-per-data ratio
- moving parts = mechanical wear out
- btw: Spinning up is most stressful for the disk. Avoid unnecessary up/down cycles.
- Lots of electronics…
- Faster than tape, slower than SSD

SSD:

- Super fast!
- (yet still) more expensive than HDDs (equal size)
- NO moving parts: good!
- Lots of electronics though…
- Limited number of writes, but:
- Not as relevant for content that is written once, then stays (=archival use case)

Good to know:

- Higher data density = more impact of an error. Example: 1mm hole in CD vs DVD vs BluRay - or SD vs MicroSD, etc.
- HDD: The longer it is active, the shorter is lives.
- But: "*Hardware that lies, dies.*"
- Be aware if your HDD is "Shingled (SMR)" or not.

Speaker notes

Density:

- Think of a 1mm hole on an optical carrier: CD vs DVD vs BluRay
- All 3 have the same dimension, but different data density.
- When choosing a carrier: Do you really need it to be as small/dense as possible?

Hardware that lies, dies:

Got external HDDs stored on shelves?
You may want to power them up every once in a while (every 1-2 years or so), and copy the data off them before the hardware gets old. Because: Electronics fail (capacitors), lubricants harden out or movable parts get “lazy” or stuck.

Das Werkstatt Rule #2 (German): “Hardware die liegt, stirbt” Useful “Das Werkstatt” principles

Shingled HDD:

> *“The more time a shingled disk spends in band compaction, the more it is wasting resources.”*

Source: Classifying Data to Reduce Long-Term Data Movement in Shingled Write Disks This means, SMR disks are active when idle, causing them to potentially wear out faster.

Tape & Drives

LTO tape with a drive

Speaker notes

There’s more to say on drive+media storage, but I’ll focus on LTO tapes here, since they become more “important”.

What may seem obvious is, that even if a tape may last a very very long time: Without a drive, you’re pretty much lost.

LTO Generations

Imagine you find an old LTO tape…

- Not every drive can read every tape.
- New LTO generation release: ~every 2-3 years.
- < LTO-7: Read=2 gen. / Write=1 gen.
- BUT: LTO-8: Read/write=**1 gen.**!

Speaker notes

LTO improvements include new LTO generations: Higher data density at same form factor. This allows the mechanics (of drives and robots) to stay compatible, with only requiring certain parts to be updated to support a new generation.

The LTO consortium guarantees a certain level of downwards compatibility. This used to be:

- Read: 2 generations down
- Write: 1 generation down

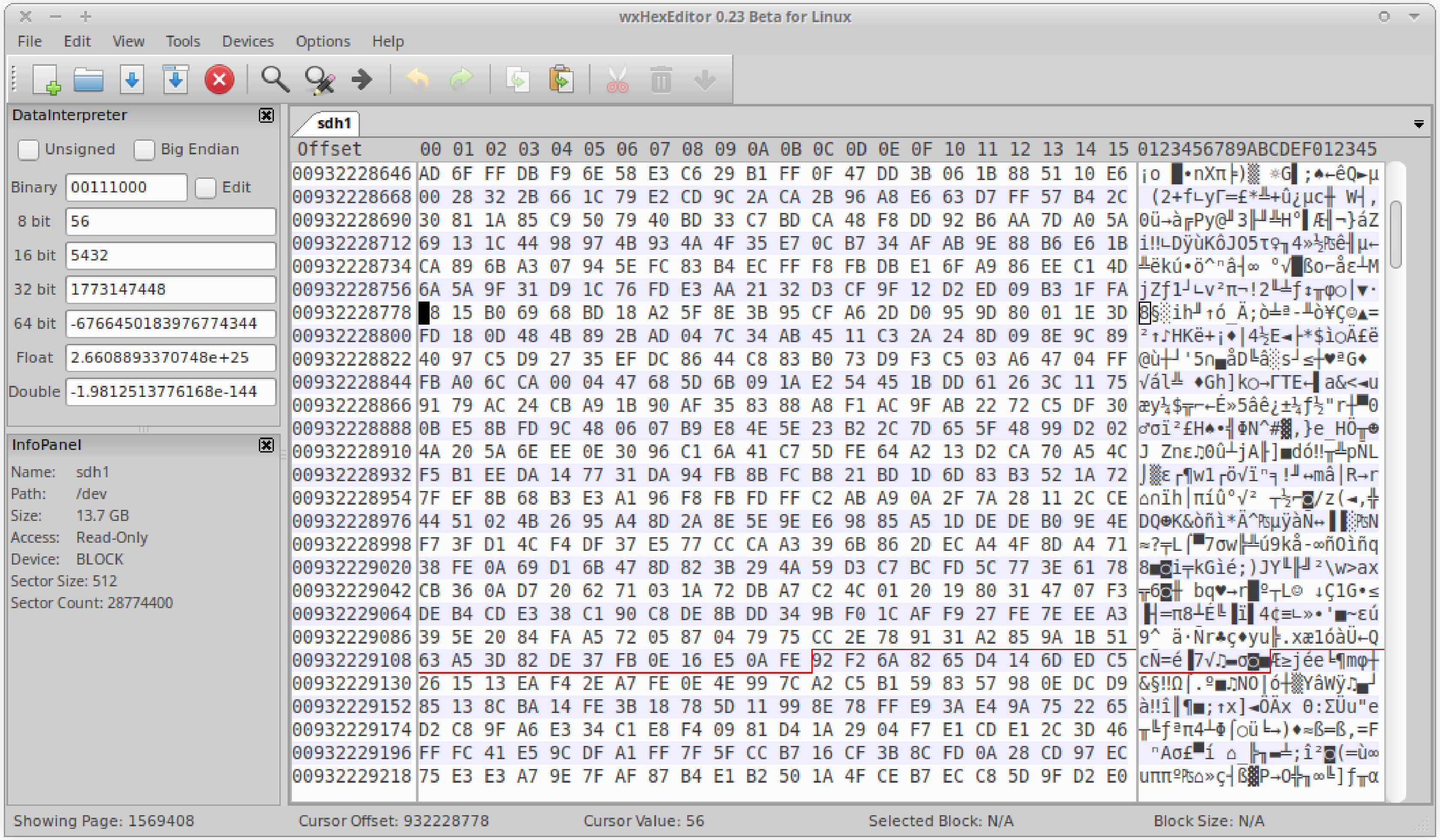
Example: LTO-6 drive should be able to read LTO 6,5,4 - but write only 6 and 5.

With LTO-8, this was reduced to 1 generation backwards compatibility, so an LTO-8 drive can *not* read LTO-6, only back to LTO-7

New LTO generation release: About every 2-3 years. This must be considered for migration!

This also means that you might want to keep a few drives of different generations in house?

The File System



```
[ ] Unsigned   [ ] Big Endian
Binary  00111000  [ ] Edit
8 bit   56
16 bit  5432
32 bit  1773147448
64 bit  -6766450183976774344
Float   2.6608893370748e+25
Double  -1.9812513776168e-144
```

```
Name:    sdh1
Path:    /dev
Size:    13.7 GB
Access:  Read-Only
Device:  BLOCK
Sector Size: 512
Sector Count: 28774400
```

```
Offset      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 0123456789ABCDEF012345
00932228646 AD 6F FF DB F9 6E 58 E3 C6 29 B1 FF 0F 47 DD 3B 06 1B 88 51 10 E6 ¡o █•nXπ╞)▒ ☼G▌;♠←êQ►µ
00932228668 00 28 32 2B 66 1C 79 E2 CD 9C 2A CA 2B 96 A8 E6 63 D7 FF 57 B4 2C  (2+f∟yΓ═£*╩+û¿µc╫ W┤,
00932228690 30 81 1A 85 C9 50 79 40 BD 33 C7 BD CA 48 F8 DD 92 B6 AA 7D A0 5A 0ü→à╔Py@╜3╟╜╩H°▌Æ╢¬}áZ
00932228712 69 13 1C 44 98 97 4B 93 4A 4F 35 E7 0C B7 34 AF AB 9E 88 B6 E6 1B i‼∟DÿùKôJO5τ♀╖4»½₧ê╢µ←
00932228734 CA 89 6B A3 07 94 5E FC 83 B4 EC FF F8 FB DB E1 6F A9 86 EE C1 4D ╩ëkú•ö^ⁿâ┤∞ °√█ßo⌐åε┴M
00932228756 6A 5A 9F 31 D9 1C 76 FD E3 AA 21 32 D3 CF 9F 12 D2 ED 09 B3 1F FA jZƒ1┘∟v²π¬!2╙╧ƒ↕╥φ○│▼·
00932228778 38 15 B0 69 68 BD 18 A2 5F 8E 3B 95 CF A6 2D D0 95 9D 80 01 1E 3D 8§░ih╜↑ó_Ä;ò╧ª-╨ò¥Ç☺▲=
00932228800 FD 18 0D 48 4B 89 2B AD 04 7C 34 AB 45 11 C3 2A 24 8D 09 8E 9C 89 ²↑♪HKë+¡♦|4½E◄├*$ì○Ä£ë
00932228822 40 97 C5 D9 27 35 EF DC 86 44 C8 83 B0 73 D9 F3 C5 03 A6 47 04 FF @ù┼┘'5∩▄åD╚â░s┘≤┼♥ªG♦ 
00932228844 FB A0 6C CA 00 04 47 68 5D 6B 09 1A E2 54 45 1B DD 61 26 3C 11 75 √ál╩ ♦Gh]k○→ΓTE←▌a&<◄u
00932228866 91 79 AC 24 CB A9 1B 90 AF 35 83 88 A8 F1 AC 9F AB 22 72 C5 DF 30 æy¼$╦⌐←É»5âê¿±¼ƒ½"r┼▀0
00932228888 0B E5 8B FD 9C 48 06 07 B9 E8 4E 5E 23 B2 2C 7D 65 5F 48 99 D2 02 ♂σï²£H♠•╣ΦN^#▓,}e_HÖ╥☻
00932228910 4A 20 5A 6E EE 0E 30 96 C1 6A 41 C7 5D FE 64 A2 13 D2 CA 70 A5 4C J Znε♫0û┴jA╟]■dó‼╥╩pÑL
00932228932 F5 B1 EE DA 14 77 31 DA 94 FB 8B FC B8 21 BD 1D 6D 83 B3 52 1A 72 ⌡▒ε┌¶w1┌ö√ïⁿ╕!╜↔mâ│R→r
00932228954 7F EF 8B 68 B3 E3 A1 96 F8 FB FD FF C2 AB A9 0A 2F 7A 28 11 2C CE ⌂∩ïh│πíû°√² ┬½⌐◙/z(◄,╬
00932228976 44 51 02 4B 26 95 A4 8D 2A 8E 5E 9E E6 98 85 A5 1D DE DE B0 9E 4E DQ☻K&òñì*Ä^₧µÿàÑ↔▐▐░₧N
00932228998 F7 3F D1 4C F4 DF 37 E5 77 CC CA A3 39 6B 86 2D EC A4 4F 8D A4 71 ≈?╤L⌠▀7σw╠╩ú9kå-∞ñOìñq
00932229020 38 FE 0A 69 D1 6B 47 8D 82 3B 29 4A 59 D3 C7 BC FD 5C 77 3E 61 78 8■◙i╤kGìé;)JY╙╟╝²\w>ax
00932229042 CB 36 0A D7 20 62 71 03 1A 72 DB A7 C2 4C 01 20 19 80 31 47 07 F3 ╦6◙╫ bq♥→r█º┬L☺ ↓Ç1G•≤
00932229064 DE B4 CD E3 38 C1 90 C8 DE 8B DD 34 9B F0 1C AF F9 27 FE 7E EE A3 ▐┤═π8┴É╚▐ï▌4¢≡∟»∙'■~εú
00932229086 39 5E 20 84 FA A5 72 05 87 04 79 75 CC 2E 78 91 31 A2 85 9A 1B 51 9^ ä·Ñr♣ç♦yu╠.xæ1óàÜ←Q
00932229108 63 A5 3D 82 DE 37 FB 0E 16 E5 0A FE 92 F2 6A 82 65 D4 14 6D ED C5 cÑ=é▐7√♫▬σ◙■Æ≥jée╘¶mφ┼
00932229130 26 15 13 EA F4 2E A7 FE 0E 4E 99 7C A2 C5 B1 59 83 57 98 0E DC D9 &§‼Ω⌠.º■♫NÖ|ó┼▒YâWÿ♫▄┘
00932229152 85 13 8C BA 14 FE 3B 18 78 5D 11 99 8E 78 FF E9 3A E4 9A 75 22 65 à‼î║¶■;↑x]◄ÖÄx Θ:ΣÜu"e
00932229174 D2 C8 9F A6 E3 34 C1 E8 F4 09 81 D4 1A 29 04 F7 E1 CD E1 2C 3D 46 ╥╚ƒªπ4┴Φ⌠○ü╘→)♦≈ß═ß,=F
00932229196 FF FC 41 E5 9C DF A1 FF 7F 5F CC B7 16 CF 3B 8C FD 0A 28 CD 97 EC  ⁿAσ£▀í ⌂_╠╖▬╧;î²◙(═ù∞
00932229218 75 E3 E3 A7 9E 7F AF 87 B4 E1 B2 50 1A 4F CE B7 EC C8 5D 9F D2 E0 uππº₧⌂»ç┤ß▓P→O╬╖∞╚]ƒ╥α
```

```
Showing Page: 1569408   Cursor Offset: 932228778   Cursor Value: 56   Selected Block: N/A   Block Size: N/A
```

A drive partition viewed as raw data

Speaker notes

That image shows a disk drive partition as quasi-uninterpreted raw data. This is what your folders and files would look like without a “file system.”

Ever encountered an “unreadable” storage medium? (SD card, USB stick, HDD, etc?) In most cases the reason were broken bits in its Filesystem. The data is often still where it was.

A File system (FS) is what forms raw data into a file/folder structure

- Storage without FS = Byte pulp. Like pages of a phone book torn apart, page numbers and index removed - then shuffled.
- Filesystem = A data format Therefore, the same requirements for preservability/accessability apply:
    - Is it documented?
    - Who can read it?
    - Under which terms and conditions?
- Different FS = different formats

Popular File Systems

- FAT{16,32} (Microsoft)
- NTFS (Microsoft)
- EXT{2,3,4} (Linux)
- **ZFS** (Sun)
- HFS+ (Apple)
- **LTFS** (IBM)

Speaker notes

FAT:

- FAT introduced in 1977.
- Widespread, well-known and documented
- Default for SD cards and embedded devices
- FAT32: 32bit limitation (max 4GB filesize, limited max. device size)
- No access permissions (=everyone can read/write everything)
- Very simple and straightforward filesystem
- Non-free / license fees by Microsoft.

NTFS:

- Introduced in 1993.
- max volume size (different implementations): 256 TiB (pre 2016), now 8 PiB (2019 or later)
- Non-free / license fees by Microsoft.

EXT:

- ext2 introduced in 1993.
- ext4 standard for Linux-based OS.
- ext4 and NTFS supported by digital cinema standard. (Because the projectors probably run Linux ;))
- max volume size: 1 EiB (1024 PiB)
- No licensing fees

ZFS: * Introduced in 2005 * More than “just a filesystem”: combines RAID (redundancy, handle harddisk failure) Logical Volume Management (LVM) and filesystem in one. * Integrated integrity checks * (recently) well known among IT administrators of unixoid systems. * Requires slightly more experience to handle (because of complexity. * Very interesting filesystem (also for preservation). Too much to describe here. You may want to read more on Wikipedia :)

HFS+: * HFS Introduced in 1985. * max volume size: 8 EB * Successor of “HFS” * Officially only supported on Apple OS

LTFS: * LTFS first prototype (Linux and Mac OS X) during 2008/2009 * Only for LTO tapes. * See separate page/slide on LTFS for details.

LTFS: Linear Tape File System

- Open specification = vendor neutral
- Better for preservation, but may not support “convenience” features.
- All implementations must:
    - Correctly read media that was compliant with any prior version.
    - Write media that is compliant with the version they claim compliance with.

Speaker notes

The Linear Tape File System (LTFS) is a good solution to avoid vendor lock-in by tape filesystems. Please make sure that you can read the data written onto the tapes under the following conditions:

- Different drive
- From different vendor
- With (FOSS) software (and/or from different vendor)

Otherwise, you won’t really know if you can access your data as technology- and vendor-neutral as desired.

By default, different applications may write data in their own way on LTO tapes. This allows them to implement some non-standard features that often provide more “convenience” by offering additional features. Using such vendor-specific features, instead of LTFS has the downside of reducing the possibilites to read that data under different conditions in the future.

Therefore it is advised to use LTFS for long term preservation use cases.

File system: Disaster relevant?

- Deleted files are still there (maybe fragmented).
- Different FS = different error resilience and recovery options.
- Does it scale? (moving files is when they're most vulnerable...)
- Tools/knowhow to deal with recovery of broken filesystems in your setups?
- Logical Volume Management (LVM) snapshots

Speaker notes

- Deleted files: Deleting files is just flagging them in the index and their disk space is marked as available again. The data is still where it was - until a new write operation (=creating new data) uses the same space and overwrites the old data.

  Related articles:
  - Get Your Data Back with Linux-Based Data Recovery Tools
  - [How to Install and Use TestDisk Data Recovery Tool in Linux]((https://www.tecmint.com/install-testdisk-data-recovery-tool-in-linux/)
  - Scalpel: Extract lost files from disk/card image

- Different FS have different error resilience and recovery options: Go ask your IT guys! ZFS is pretty awesome, but “simple and well known” is also nice for recovery.

- FS recovery requires understanding the FS format. Got documentation? Source code?

- Logical Volume Management (LVM) Remember VM-snapshots? You can do the same on storage level (below filesystem): LVM Snapshots

- Does it scale?
  - Total number of files/folders?
  - Length of path?
  - Suitable for NAS/SAN/cluster?
  - FS vs Blocks vs Object-based?

Linking Metadata and Content

- Metadata: Database (DB)
- Content: Storage
- Connection: By identifier and/or path. Stored in DB.

File-naming and structure

1. `Høvegaard, Björn Clément/16mm 571,1 Jahr 1978 - Mågst im Wald tanz’n - MPEG-4 ProRes.mov`
2. `f23bfeb9-7558-4a9c-bfb4-dd2d5a0409de.mxf`
3. `VX/00/VX-00815/vx-00815.mkv`

Speaker notes

- Explain pros/cons of the naming examples.

Notes on the above examples:

1. Based on an actual real-world filename.

    - DON'T! This is one of the worst cases ever!
    - Big problems dealing with these files! (Access, conversion, migration, etc)
    - Do: Stay with alphanumeric characters.
    - Do: Avoid spaces.
    - Don't: Codec/format in preservation filenames. It will change! And then it'll be wrong. If you fix the filename: All references to it will be invalid.
    - Don't: Titles will contain bad characters at some point.

2. UUID

    Just an example, what a filename generated by a DAM/MAM may look like. It's uniquely identifiable by a system, but a bit hard for humans. This is fine, as long as the data to correlate your files with your metadata is intact.

3. Example of naming persistency

    The signature syntax is taken from the Austrian Mediathek. The identifier (Example: VX-00815) is short enough to be memorized by a human, and has a syntax that allows to identify the media type somewhat (V=VHS, VX=other video).

    The path/filename structure also has a syntax rule that is used to split up the sub-directories, according to the item's signature. This allows to avoid too many items/files in a single folder, but in a way that human (administrators) can still deal with the structure.

The catalogue/DAM knows the identifier and can then link to the files on the storage, according to these rules.

This also allows access to the files outside and independent of the catalogue/DAM, for any other reasons. Popular use cases are: automated format migrations, digitization ingest systems from different vendors/different media types.

It does have some drawbacks and doesn’t scale infinitely, but it’s a nice example that allows tool/OS/filesystem/vendor agnostic operations and has survived several migrations (tool/OS/filesystem/vendor) successfully.

Dos, Donts, Letgos

- **Do**:
  - Naming persistency, syntax rules
- **Don’t**:
  - Non-alphanumeric characters + spaces
  - Codec / format names
  - Title, Author, etc.
- **Let go**:
  - Human graspable names…

Speaker notes

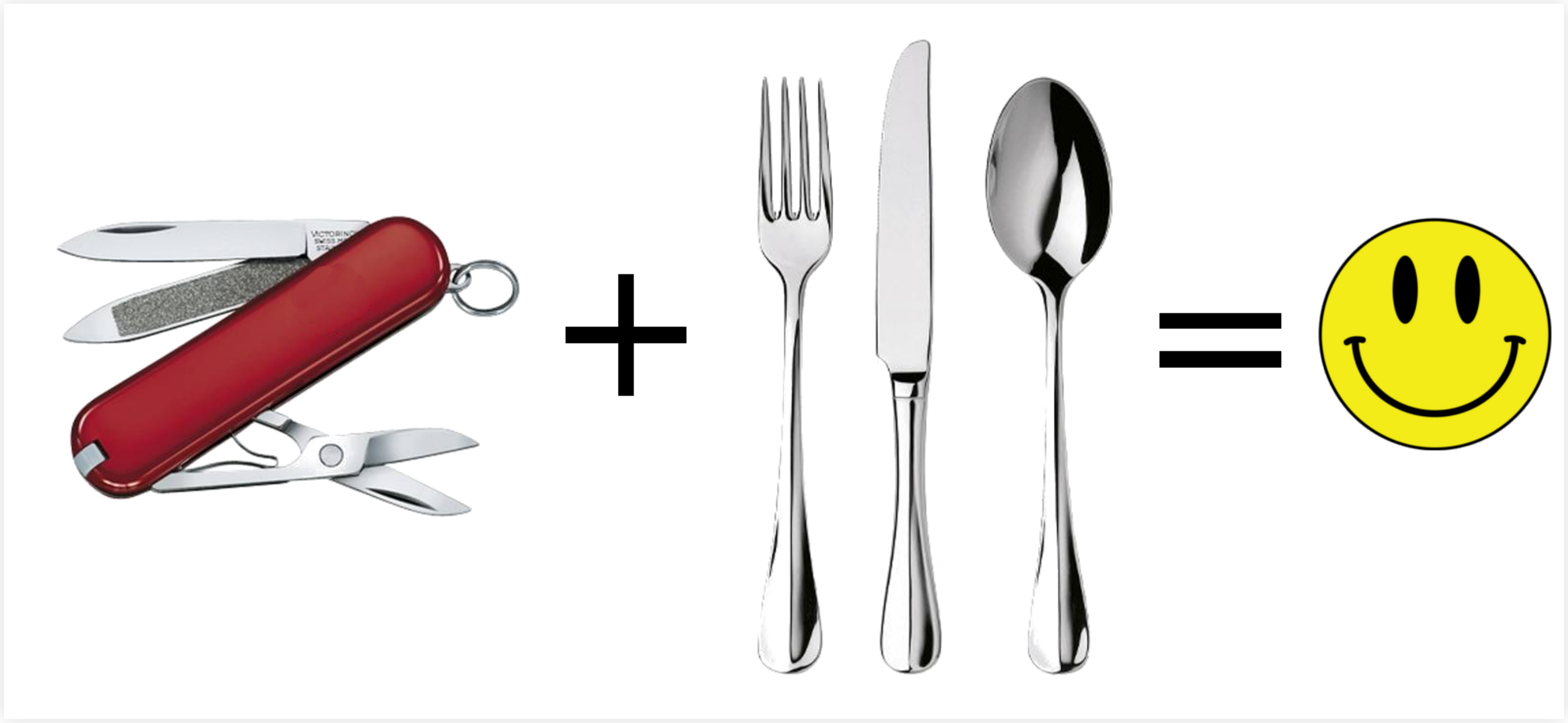
File-naming and structure

- Relevant for:
    - Tool/OS/Filesystem agnostic operation
    - Migration
    - Disaster recovery: find and assign files to their catalogue metadata.
- Most important:
    - **Naming persistency:** "Ensure the persistence of the originally-applied file name. Trying to rename files causes all sorts of administrative and technical problems and, if not handled careful, can lead to synchronization errors between a file and and the external database record that holds descriptive information about the file."
    - **Naming syntax rules:** No rule/system means that all files will have to be linked quasi manually with a catalogue/DAM. This has so many drawbacks and causes so many issues, I don't even know where to begin.
    - Avoid problematic characters (and spaces!). Seriously.
- Nice to know? "Detox" "*Detox is a utility designed to clean up filenames. It replaces difficult to work with characters, such as spaces, with standard equivalents. It will also clean up filenames with UTF-8 or Latin-1 (or CP-1252) characters in them.*"
- Let go: Beyond a certain number of assets, the names will have to be generated by rules, not human-readable anymore. This can feel very uncomfortable, because you lose control. You have to let go. Many DAMs auto-generate the filenames and folder structure. But this means: Your files are now *owned* by the DAM (vendor). Do you have plan what happens if you and the DAM go separate ways?
- Possible consequences:
    - lose MAM/DB = lose order = chaos

Files intact, but MAM gone?

Needle in a haystack?

Store identifying metadata



Speaker notes

The following may improve finding and/or re-organizing your files independent of the asset management system that has been used to allocate them before:

Store some identifying metadata in+next to your files:

- Minimalistic inside the file: Textual data that is most likely *not* to change over the lifetime of the file. Good practice: title, identifier, institution name, digitization date

- More metadata next to the content files (sidecar).
    - Store images as images.
    - Store textual data as structured text (XML is text). Makes it easier to modify, augment and migrate this data, but leave the large files in place.

- As simple as possible, as complicated as necessary.

All in one?

“All” can sometimes be too much…

Speaker notes

All-in-one file?

Sometimes nice, but consider the following before you do that:

- More data inside the file = Requires more complex data format (container/codec)
- This limits the number of applications that fully support these files (access/write).
- May make access and/or format migrations harder or more error prone.
- Augmenting/modifying/correcting data inside a file will change its hashcode. And depending on the format, it requires to read/write the whole file, even if you change just a single letter!

Failsafe mechanisms

Just to be sure.

S.M.A.R.T.

Self-Monitoring, Analysis and Reporting Technology

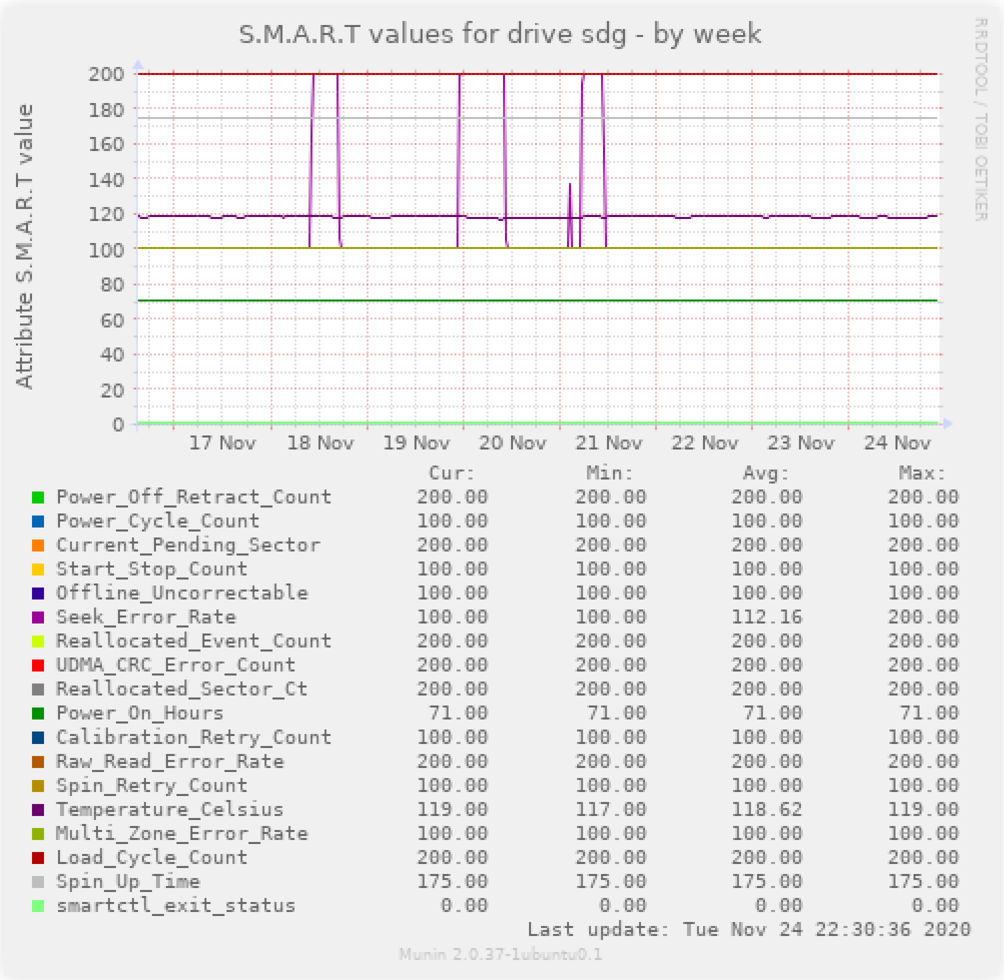
“[…] is a monitoring system included in computer hard disk drives (HDDs), solid-state drives (SSDs),[1] and eMMC drives. Its primary function is to detect and report various indicators of drive reliability with the intent of anticipating imminent hardware failures.”

Source: Wikipedia: S.M.A.R.T.

Speaker notes

- Part of the ATA standard since around 2004.
  - All HDDs these days usually support SMART.
  - Different vendors/drives return different readings.
  - “The meaning and handling of the raw value is a secret of the vendors embedded S.M.A.R.T.-Software on the disk.”
  - It makes sense to monitor these values and alert if values deviate from a certain threshold.

S.M.A.R.T. Graph



		Cur:	Min:	Avg:	Max:
🟩	Power_Off_Retract_Count	200.00	200.00	200.00	200.00
🟦	Power_Cycle_Count	100.00	100.00	100.00	100.00
🟧	Current_Pending_Sector	200.00	200.00	200.00	200.00
🟨	Start_Stop_Count	100.00	100.00	100.00	100.00
🟪	Offline_Uncorrectable	100.00	100.00	100.00	100.00
🟪	Seek_Error_Rate	100.00	100.00	112.16	200.00
🟨	Reallocated_Event_Count	200.00	200.00	200.00	200.00
🟥	UDMA_CRC_Error_Count	200.00	200.00	200.00	200.00
⬛	Reallocated_Sector_Ct	200.00	200.00	200.00	200.00
🟩	Power_On_Hours	71.00	71.00	71.00	71.00
🟦	Calibration_Retry_Count	100.00	100.00	100.00	100.00
🟫	Raw_Read_Error_Rate	200.00	200.00	200.00	200.00
🟨	Spin_Retry_Count	100.00	100.00	100.00	100.00
🟪	Temperature_Celsius	119.00	117.00	118.62	119.00
🟩	Multi_Zone_Error_Rate	100.00	100.00	100.00	100.00
🟥	Load_Cycle_Count	200.00	200.00	200.00	200.00
⬜	Spin_Up_Time	175.00	175.00	175.00	175.00
🟩	smartctl_exit_status	0.00	0.00	0.00	0.00

Last update: Tue Nov 24 22:30:36 2020

Munin 2.0.37-1ubuntu0.1

SMART graph for a single drive

Speaker notes

- SMART values can be collected over time and drawn as graph. This allows to literally “see” if some reading is “going down”.
    - Generated using Munin, which uses the great RRDTool to generate graphs.

RAID

Redundant Array of Inexpensive Disks

"[...] is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both."

Source: Wikipedia: RAID

Speaker notes

- Good ‘old’ friend: exists since before 1988
    - Here you see a classical hardware RAID chassis.
    - This is just one option.

RAID

- Different RAID levels:
  - Stripe = Fast, but dangerous!
  - Mirror = Perfect for OS system disks
  - RAID5/6 = 1 or 2 disks fault tolerance (parity bits)
- RAID is *not* a backup.

Speaker notes

Data replication is not a Backup: It is not unusual to discover an undesired modification or error only later on.

- Sync:
    - File changes (new, modification) are immediately replicated
    - Unwanted actions, too! (wrong modification or deletion)
- Async:
    - Allows you to go back in time.
    - Allows multiple copies/revisions of the same data.
    - “Long” time between new/modification and the copy.

Hard- vs Software RAID?

- **Hardware:**
  - “easier” to setup and swap disks.
  - Risk of vendor lock-in.
  - Caution: Mainboard controllers are often “Pseudo-RAID”
- **Software:**
  - More complex to set up and swap disks.
  - Linux kernel softraid: Very portable!
  - Speed concerns with modern hardware possibly neglectible.

Speaker notes

Linux mdadm RAID:

- Easily portable/compatible/reconstructable
- Well documented.
- Production stable for many years.

Windows software RAID:

- Surprising, but it seems to be quite undocumented. There seem to be no official data recovery tools (or instructions) provided by Microsoft.
- I’d only use it to mirror devices.

Proprietary hardware/firmware RAID issue:

Often proprietary data layout, which does not guarantee that the disks can be read in a device/setup by another vendor - or different firmware/model.

If you don’t have matching spare parts on-site, your **downtime** is until you get one. If the unit was older than, let’s say 3 years, it’s not even sure if you get a compatible replacement anymore. This often leads to a forced upgrade of the whole hardware RAID.

Hardware RAID:

- Requires hardware controller. Sometimes it’s an embedded external system.
- “Easier” to configure and swap disks
- No impact on the host CPU/RAM, since everything’s handled in the RAID hardware.
- Completely transparent to the host. “Feels” like a physical disk.
- Used to be necessary, before PCs got fast enough for Software RAID.
- Host cannot access the individual drives directly (e.g. cannot access SMART readings)
- Uses the whole disk.

Software RAID:

- Uses CPU/RAM of the host machine.
- Flexible configuration, mixed devices.
- Can be made on partition level.
- Host can access the individual drives directly (e.g. allows SMART monitoring, etc)
- Linux kernel: open, documented and portable data layout = plug and play on any hardware.

More detailed comparisons:

- Hardware raid or software raid? Our take on the great religious debate (2012)
- Update: Hardware vs Software RAID: The great debate. (2013)
- https://www.cyberciti.biz/tips/raid-hardware-vs-raid-software.html (a bit aged, 2009)

Sad, but true:

- With > 4TB disks, RAID-6 may be insufficient…
- Long rebuild times may “burn out” the left-over disks.
- ZFS supports 3 disks parity, but … future?

Speaker notes

The algorithm RAID-6 uses to provide data redundancy for rebuilding in case of harddisks failing, may have come to an end with HDD capacities beyond about 4TB per-disk.

The reason for that is, that in case of a disk failure, the RAID mechanism reads the parity data from all remaining disks to reconstruct the one that “has gone”. With larger disks, the time for this rebuild to happen is getting longer. During that period, all remaining disks are constantly “stressed” until the required data has been read and the new disk rewritten.

Considering that disk failures more likely occur in RAID arrays that have been running for quite some time, it can be assumed that the other disks may be stressed too hard during that rebuild, causing other disks to fail *due* to the recovery - resulting in a complete data loss.

Cluster & replication

- Multiple copies on multiple servers (“cluster nodes”)
- Synchronized (replicated) automatically
- Off-site replication possible
- Replication is *not* a backup!
- What if a node goes down?
- How does “the cloud” do it?

Speaker notes

Where RAID provides failover redundancy for individual storage devices, a storage cluster may provide failover redundancy by means of whole PCs.

Each PC is called a “node” and is designed as a component that may fail, while the rest of the cluster can maintain their services. This is how cloud storage/service providers provide transparent failover setups.

Replication is not a backup.

Replication mechanisms try to keep all their data in-sync as quickly and seamlessly as possible. This means that if you delete a file, it is usually not known by any replicating system if that was on purpose on in error.

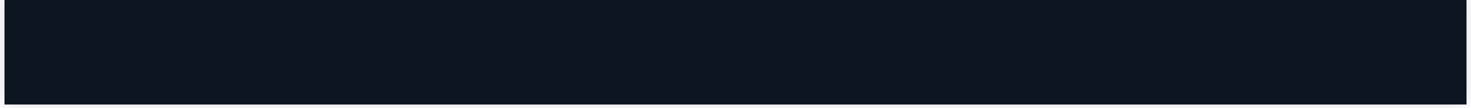
Therefore, the replication mechanism happily will synchronize this change (=the deletion) to all of its nodes. Voila.

You get it, why this would be a bad backup option? ;)

There is NO CLOUD, just

other people's computers

fsfe.org



Speaker notes

The term “Cloud” in computing came from marketing, and describes nothing else than digital services that already existed for over 20 years - but now had a nicer GUI and focus on convenient usability.

There’s “cloud storage” (e.g. Owncloud/Nextcloud, Dropbox, etc) and “cloud computing/services” (GDocs, etc).

In this context we will only deal with “cloud storage”.

Storage Debate!

Speaker notes

Make 2 groups:

- One group for “all in-house”
- One group for “all external”
- Think of 5 “pros” why your scenario is a good choice.
- After 7 min. we’ll get back and debate.

Inside = The same stuff

- Combining a mix of most (if not all) the above mentioned technologies.
- Mostly unix-like operating systems (e.g. Linux).
- And lots of carriers.
- maintaining it = other people's problems.

So what if cluster nodes fail?

- Another node takes over.
- Seamlessly. Transparently.
- The faulty node is removed & replaced.
- Life continues.

Speaker notes

In case a node fails, another node jumps in to take over its job(s). The failing node is replaced in due time. Comparable to RAID setups, but on another abstraction layer.

Of course they need some kind of “magic” (=algorithms, data protocols, mechanisms, applications, etc) to do so.

There are many different cluster operating systems and applications, some of

Erasure Coding

> *“protects data from multiple drives failure, unlike RAID or replication. For example, RAID6 can protect against two drive failure whereas in MinIO erasure code you can lose as many as half of drives and still the data remains safe.”*

Source: MinIO Erasure Code Quickstart Guide

> *“Compared to data replication, erasure-coding approaches have better performance at reducing storage redundancy and data recovery bandwidth.”*

Source: Reliability Assurance of Big Data in the Cloud (2015)

Speaker notes

Erasure coding is a different (algorithmic) approach to provide data redundancy to counteract device/node failures.

Quote from the Minio website:

"Erasure code is a mathematical algorithm to reconstruct missing or corrupted data. MinIO uses Reed-Solomon code to shard objects into variable data and parity blocks. For example, in a 12 drive setup, an object can be sharded to a variable number of data and parity blocks across all the drives - ranging from six data and six parity blocks to ten data and two parity blocks.

By default, MinIO shards the objects across N/2 data and N/2 parity drives. Though, you can use storage classes to use a custom configuration. We recommend N/2 data and parity blocks, as it ensures the best protection from drive failures.

In 12 drive example above, with MinIO server running in the default configuration, you can lose any of the six drives and still reconstruct the data reliably from the remaining drives."

More math at: Wikipedia: Erasure Code

Quote from Wikipedia:

"Example: In RS (10, 4) code, which is used in Facebook for their HDFS,[4] 10 MB of user data is divided into ten 1MB blocks. Then, four additional 1 MB parity blocks are created to provide redundancy. This can tolerate up to 4 concurrent failures. The storage overhead here is 14/10 = 1.4X.

In the case of a fully replicated system, the 10 MB of user data will have to be replicated 4 times to tolerate up to 4 concurrent failures. The storage overhead in that case will be 50/10 = 5X.

This gives an idea of the lower storage overhead of erasure-coded storage compared to full replication and thus the attraction in today’s storage systems."

Erasure Coding

- More parity possible than RAID
- Data can be spread across network cluster nodes.
- Does not suffer from rebuild “burn out”.
- More complex to set up & compute.
- Not that widely known/supported/used yet.

Speaker notes

Links:

- MinIO: Storage usage
- MinIO: Erasure Code Quickstart Guide
- Erasure coding vs RAID: Data protection in the cloud era (ComputerWeekly)
- RAID Vs. Erasure Coding (NetworkComputing)
- What is erasure coding? (VirtuService)
- Isilion.com: Erasure Coding (Isilon)
- Erasure Coding vs. RAID: Which Is Right and When? (ITProToday)
- Wikipedia: Erasure code

Object storage

- Files become “objects”.
- There are no folders (as we’re used to)
- There’s just an identifier.
- Arbitrary metadata may be assigned to each “object”.
- Scales infinitely (theoretically).
- Clusters of block storages with a clever abstraction layer.

Speaker notes

You may have been using object based storage on Google Drive or Amazon S3.

Pretty cool, but it handles very differently from any regular file/folder based storage! Please do your homework before deploying this.

It is still pretty new and if done wrong, it can cause more troubles/effort than a regular, hierarchical FS storage.

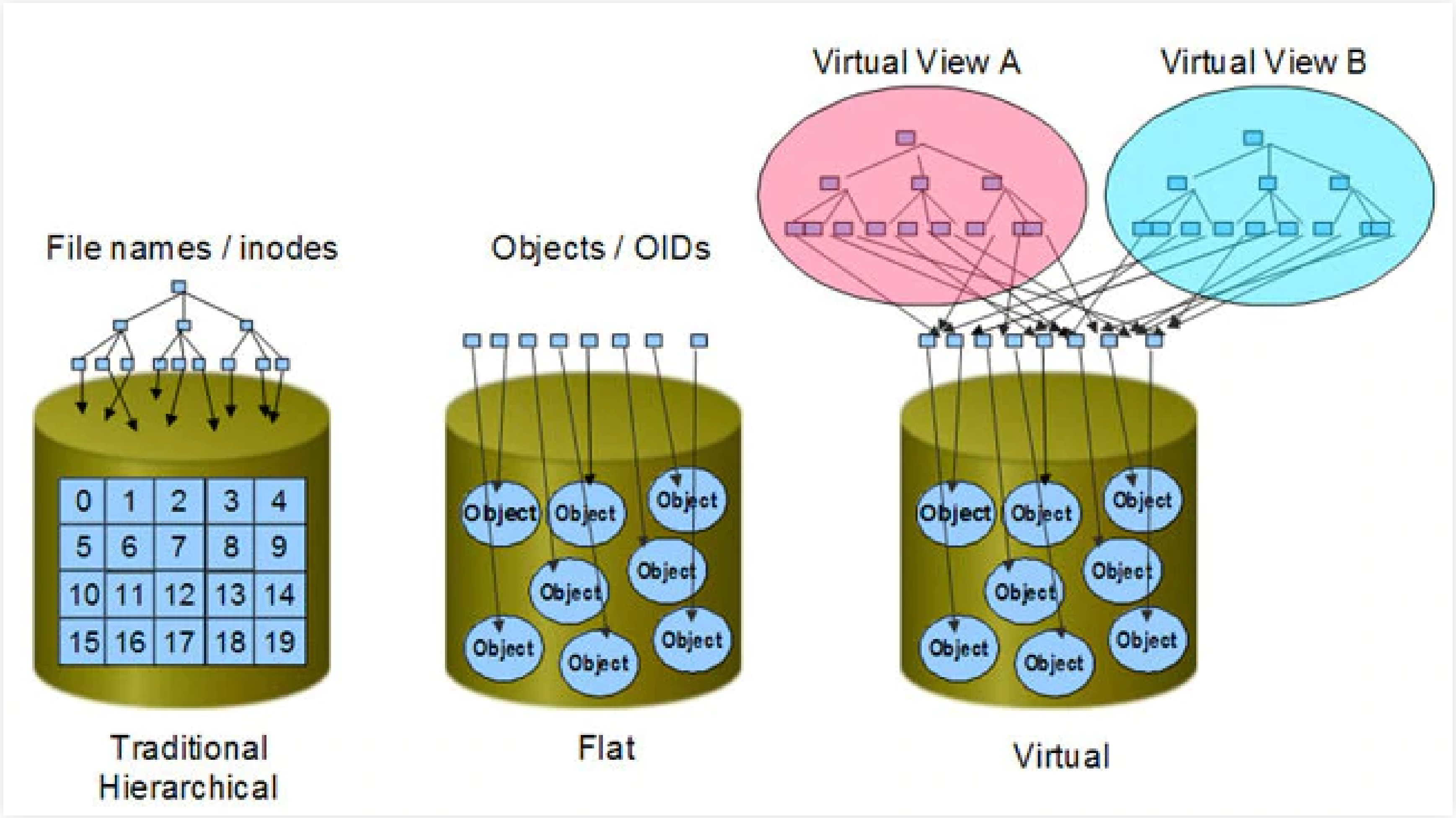
What is often left out by marketing:

- Underneith, it is still using block storage.
- Just with clever abstraction layer(s).
- Important: Your toolchain needs to be object-storage-aware! Regular, well-known tools cannot easily access object files as it used to be.
- I mention it here, because if done right, it scales.

I’m pretty sure, this is the future - and (at least large scale) storage will be going from classical (filesystem-based) to this (=object based) way.

Object storage

Beyond classical, hierarchical filesystems

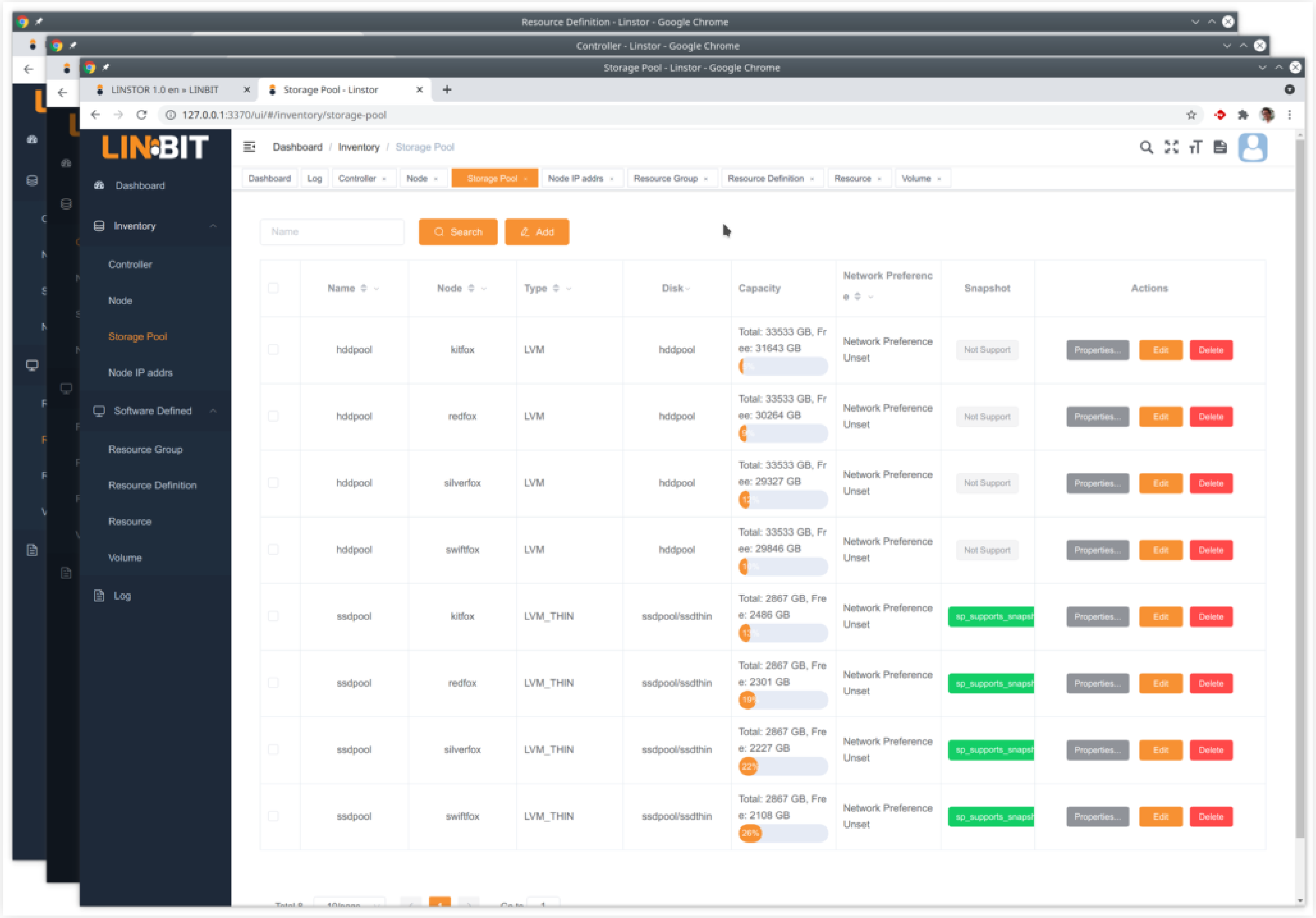


Object storage = The future?

Object based storage may very likely replace hierarchical filesystems, but things need to be rewritten/adapted to properly support it.

It’s not yet plug-compatible with existing programs.

Software Defined Storage (SDS)



	Name	Node	Type	Disk	Capacity	Network Preference	Snapshot	Actions
	hddpool	kitfox	LVM	hddpool	Total: 33533 GB, Free: 31643 GB	Network Preference Unset	Not Support	Properties... Edit Delete
	hddpool	redfox	LVM	hddpool	Total: 33533 GB, Free: 30264 GB	Network Preference Unset	Not Support	Properties... Edit Delete
	hddpool	silverfox	LVM	hddpool	Total: 33533 GB, Free: 29327 GB	Network Preference Unset	Not Support	Properties... Edit Delete
	hddpool	swiftfox	LVM	hddpool	Total: 33533 GB, Free: 29846 GB	Network Preference Unset	Not Support	Properties... Edit Delete
	ssdpool	kitfox	LVM_THIN	ssdpool/ssdthin	Total: 2867 GB, Free: 2486 GB	Network Preference Unset	sp_supports_snapsh	Properties... Edit Delete
	ssdpool	redfox	LVM_THIN	ssdpool/ssdthin	Total: 2867 GB, Free: 2301 GB 19%	Network Preference Unset	sp_supports_snapsh	Properties... Edit Delete
	ssdpool	silverfox	LVM_THIN	ssdpool/ssdthin	Total: 2867 GB, Free: 2227 GB 22%	Network Preference Unset	sp_supports_snapsh	Properties... Edit Delete
	ssdpool	swiftfox	LVM_THIN	ssdpool/ssdthin	Total: 2867 GB, Free: 2108 GB 26%	Network Preference Unset	sp_supports_snapsh	Properties... Edit Delete

Example screenshot (Linbit SDS)

Speaker notes

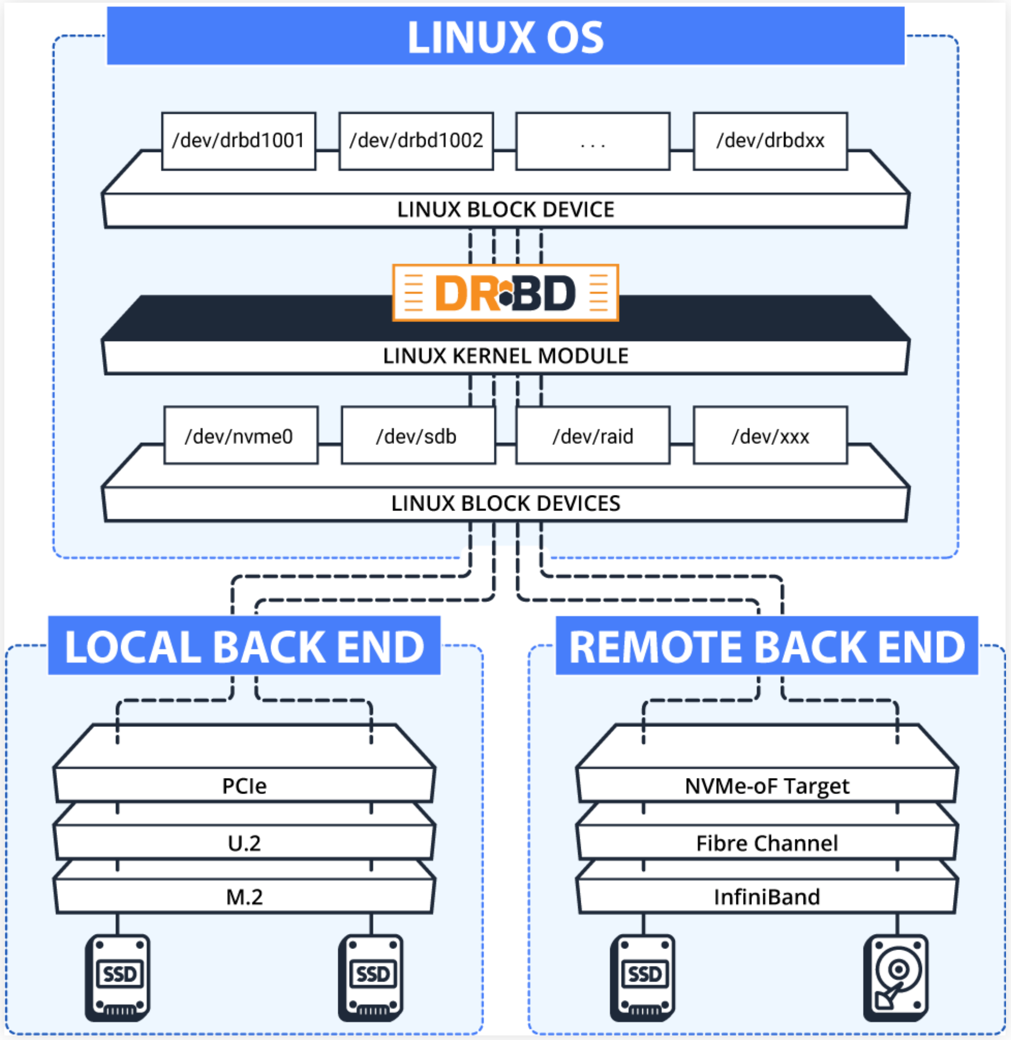
Quote from Wikipedia “Software-defined storage”:

"Software-defined storage (SDS) is a marketing term for computer data storage software for policy-based provisioning and management of data storage independent of the underlying hardware.

Software-defined storage typically includes a form of storage virtualization to separate the storage hardware from the software that manages it.

The software enabling a software-defined storage environment may also provide policy management for features such as data deduplication, replication, thin provisioning, snapshots and backup."

Again: ~~Onions~~, eh Layers.



Example of abstraction layers

Speaker notes

Quote from Linbit Website:

> *“Since the software simply presents a block device, users do not need to use complex, shared file systems. Because we can use a single node with a filesystem on top of a block device, backing disk failure is transparent to the end user.”*

Large storage that provides flexibility/scalability, the required features (distribution, replication, backup, etc, etc) - and transparency for the applications accessing the files means stacking up more layers.

Even if it ain’t rocket science, it still is far from trivial (especially setting it up, maintaining and and debugging it).

The Cloud

- No internet = no service.
- Make sure your online bandwidth is sufficient.
- Have an exit plan (+contract?) for migration.
- What if their conditions change?
- Where is your data actually stored?
- Does it matter for you? (e.g. legally)
- Again: Consider mixing.

Speaker notes

Storing data in large data centers is very convenient. All the same challenges regarding large-scale storage still apply, but they're someone else's problem now :)

What's still relevant for you though is:

- What happens if the provider changes their conditions?
- What happens in case of unstable political situations, war, etc?
- What if you want to migrate your stuff to another storage/provider?
- Does it matter (e.g. legally) if the provider could access your data?

Does it scale?

Backblaze data center

Speaker notes

The image shows one row of Backblaze storage pods in a computing center. This is just to give you an idea that:

- The Cloud literally is just a bunch of computers (and storage media inside them).
- Large volumes of data need large volumes of space. And electricity, knowhow and spare parts and staff.
- However, the underlying mechanisms and applications are designed to scale.

Does it scale?

- What if you run out of diskspace?
- Need to copy/move everything?
- Or can you simply add “more space”?
- What about your file/folder positions?
- How to know if everything is still “there” and intact?

Speaker notes

In the past with RAID, it was very common that enlarging the storage space meant: get a bigger RAID with bigger disks, then copy everything over and switch to the new storage.

With data sizes currently common for professional AV content, this approach is a very delicate one that puts your data in quite some danger, as the time where data is the most vulnerable is when it’s being moved.

Data Integrity?

Fixity information & Hashcodes :)

Hashcodes: fixed size number that’s like a fingerprint for data.

- CRC = `4294967295`
- **MD5** = `d41d8cd98f00b204e9800998ecf8427e`
- SHA256 = `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85`
- **xxHash** = `e4c191d091bd8853`

Speaker notes

Hashcodes are fixed-length numbers that are generated in a way that if even a single bit in the source data changes, that number will be completely different. They are often written in hexadecimal, therefore including the characters 0-9 and A-F.

If you have a hashcode for a set of data, it can be used to verify the bit-exact integrity of that data, by calculating the same-algorithm hashcode again and comparing them. If they're identical, the data is intact. If two distinct sets of data have the same hash, it's called a "hash collision. Hash algorithms are designed particularily to keep the chance for collision as low as possible.

Anyways: hashcodes are *a must* for safe data transfer and integrity checks.

Different hashcodes algorithms/implementations may have significantly different runtimes. When dealing with large quantities of data, this may matter.

MD5 is the most popular one around: Well known and widely supported by different applications/systems, etc.

Anyone transferring lots of uncompressed film? You may want to look at xxHash. It's designed for speed.

In case someone has heard that MD5 is broken, fear not: For plain checking of file transfer or stored data integrity, MD5 is sufficient. It was cryptographically "broken" - which is relevant for security, but not for data integrity checking.

Important: Hashcodes are not sufficient for proving authenticity! In case you need to deal with important originals/documents and you need to make sure they're originals and not forged, etc. please check out this:

- Digital Signatures
- Blockchain

Digital signatures are good, but could be signed with a date in the past (backdating).

If this is a concern, you may consider blockchain mechanisms: Blockchain cipher became popular for digital currency (like Bitcoin), but it can also be used to proof data authenticity and avoid backdating.

AFAIK there are currently no systems that implement this productively yet, but some have already researched into prototyping blockchain use for storage.

Do it early, do it continuously.

- Create hashcodes early in the lifecycle.
- Use them when transferring files.
- Storage: Ongoing integrity checks matter.
- Consider runtimes & interference with daily work.

Speaker notes

Ongoing integrity checks on your storage: Know where your collection is at.

It also makes sense to verify your backup that way.

Consider runtimes:

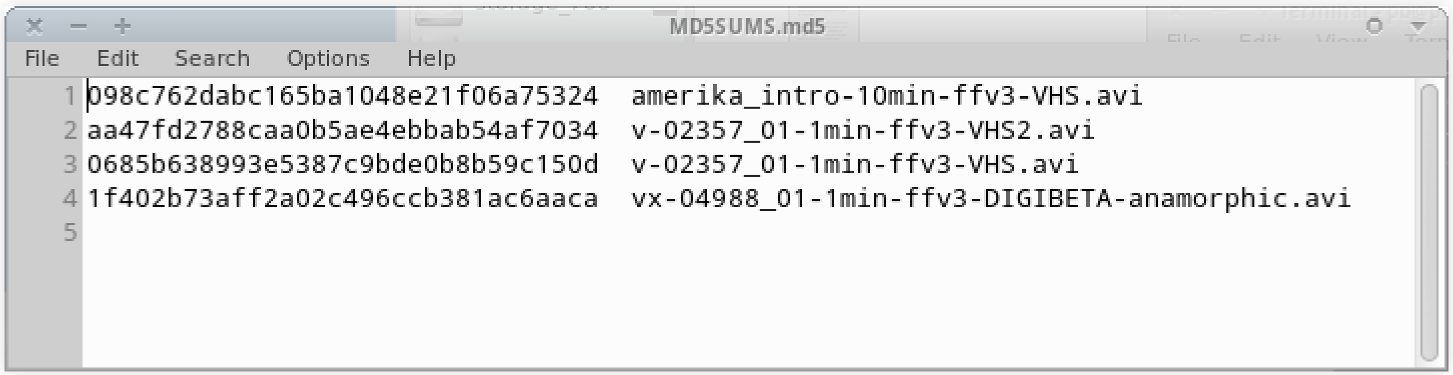
- In order to generate/verify a hashcode, the whole file must be read at least once.
- The chosen hashcode algorithm and its implementation may have impact on processing speed.
- Do some tests, calulations to find a sweet spot for how often you can do integrity checks.
- Consider throttling throughput data to avoid interference with daily work.

Real world example:

1. Archival Institution had hashcodes (ingest system created them), but never used them. They told me: “We don’t need to verify hashcodes, we have no problems” So I checked. More than 1000 items had invalid hashcodes. Since they were never checked before, it is unknown when or what caused them - and also their impact on the material.

It’s good practice to document the result of integrity checks somewhere. It helps finding issues in your setup, or at least narrowing things down.

Hashcode manifest

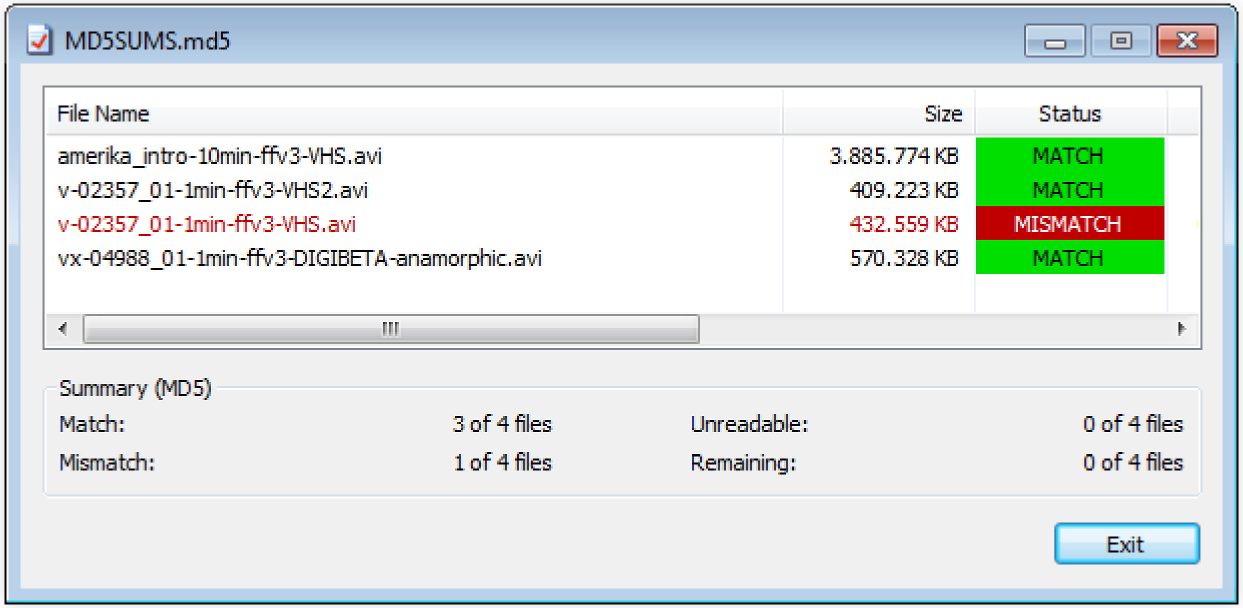


```
098c762dabc165ba1048e21f06a75324  amerika_intro-10min-ffv3-VHS.avi
aa47fd2788caa0b5ae4ebbab54af7034  v-02357_01-1min-ffv3-VHS2.avi
0685b638993e5387c9bde0b8b59c150d  v-02357_01-1min-ffv3-VHS.avi
1f402b73aff2a02c496ccb381ac6aaca  vx-04988_01-1min-ffv3-DIGIBETA-anamorphic.avi
```

As simple as that. Plain text.

Example for an MD5 file manifest

Using (=validating) hashcodes



MD5SUMS.md5

File Name	Size	Status
amerika_intro-10min-ffv3-VHS.avi	3.885.774 KB	MATCH
v-02357_01-1min-ffv3-VHS2.avi	409.223 KB	MATCH
v-02357_01-1min-ffv3-VHS.avi	432.559 KB	MISMATCH
vx-04988_01-1min-ffv3-DIGIBETA-anamorphic.avi	570.328 KB	MATCH

Summary (MD5)

Match:	3 of 4 files	Unreadable:	0 of 4 files
Mismatch:	1 of 4 files	Remaining:	0 of 4 files

HashCheck: GUI tool showing data validation

Speaker notes

Some tools

- Fixity (AVP)
- HashCheck
- QuickHash GUI
- ...

Our storage has integrity checks built in!

- Good! □
- But it can only do so as soon as data has arrived.
- Did you verify that the data got there intact? 🧐

Speaker notes

There are storage systems that have (continous) integrity checks built in. This is very good! However, it still seems to be common to simply copy files there without checking if they have ever arrived without transfer errors.

If the whole transfer chain was fine, no problem - if there was any read/transfer error: Your storage could not tell you.

If a hashcode already exists at the source, you can use this to verify bit-proof transfer.

Exercise

- Make groups of 2-3 people each.
- Download the fixity example zip:
  http://download.av-rd.com/pub/exercises/fixity/hashcode-exercise_1a.zip
- Extract the files.
- Compare the manifests with the “reference” and try to spot issues (=differences).
- Try to find out what caused these issues.
- You may want to use a “diff” tool (e.g. MELD)

Backups

The 3-2-1 Backup Rule

- Keep at least **three** copies of your data.
- Store **two** backup copies on different devices or storage media.
- Keep at least **one** backup copy offsite.

Speaker notes

That’s not a law, it’s a rule :) It’s good however to try to implement as much of it as you can.

If you can only do “2-1”, it’s better than not trying at all.

Backup integrity?

- btw: Is your backup copy (still) intact...?
- Did you check? ;)
- Tape: Backup drives on shelf vs robot?
- Handout: Backup checklist

Storage: Summary

- Filenames matter.
- There's more than just *one right* solution…
- Mixing is usually a good idea.
- RAID6 might be at its limits :(
- Consider LTFS and LTO generations
- Embrace hashcodes!
- Avoid black boxes.
- Have a backup (plan)

- fin -

Questions? Comments?

Peter Bubestinger-Steindl

`PB @ AV-RD.com`

Storage Terms Collection

- S.M.A.R.T. Self-Monitoring, Analysis and Reporting Technology
- NAS Network Attached Storage
- SAN Storage Area Network
- RAID Redundant Array of Inexpensive Disks
- Object Storage
- JBOD Just a Bunch Of Disks
- SDS Software Defined Storage

Speaker notes

You don’t have to become a storage expert, but It’s good to have heard these terms and roughly know what they mean.
This selection of terms is by faaar not complete, but these are quite likely to appear in a DLTP context.