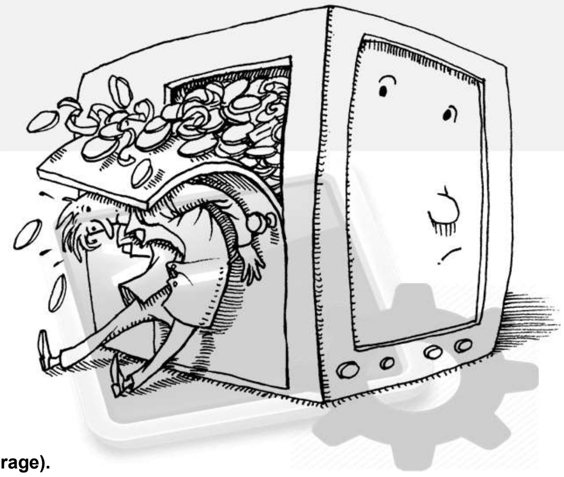
Advanced File Systems (Storage)



Reference Material:
[1] UNIX and Linux System Administration Handbook, Chapter 20 (Storage).

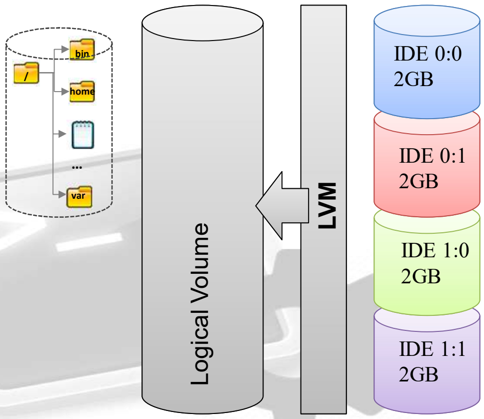
Index

Built on top of previous FS concepts (T2)

- Introduction
- Adding a new Storage Device
- File System Consistency
- Managing Filesystem
- File System Security (disk encryption)
- **Advanced Device Management**
  – Logic Volume Management (LVM)
  – Redundant Array of Inexpensive Disks (RAID)
- **Backup**

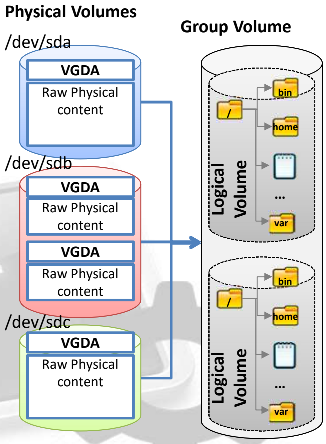
Logical Volume Manager (LVM)

- My File System has a size of 4GB, but I only have 2GB Disks. Is there any solution?
- **LVM:** creates an abstraction layer over the physical storage, allowing the creation of logical volumes(“hide” the underlying HW, exposing a single Volume to the SW).
  - **Flexible management of disk storage**: avoid the limitations imposed by disk physical size. A File System can be extended through multiple disks.
  - **Re-sizeable Storage**: logical volumes can be extended/reduced in a simple way. Some operation do not require File System umounting.



LVM Hierarchy

- **Physical Volumes** (PV)
  – Lowest level of LVM hierarchy
  – Complete disk or partition (or RAID device)
  – Contains VGDA (Volume Group Descriptor Area) and the raw physical content.
- **Group Volumes** (VG)
  – equivalent to “super-disks”
  – Built with one or more PVs
    - more PVs can be added to the GV without modifying the previous ones
- **Logical Volumes** (LV)
  – Equivalent to “super-partitions”
  – File Systems are created on a Logical Volume



LVM Administration (lvm2)

Entity	Operation	Command
PV	Create	pvcreate
	Inspect	pvdisplay
	Modify	pvchange
	Check	pvck
VG	Create	vgcreate
	Modify	vgchange
	Extend	vgextend
	Inspect	vgdisplay
	Check	vgck
	Enable	vgscan
LV	Create	lvcreate
	Modify	lvchange
	Resize	lvresize
	Inspect	lvdisplay

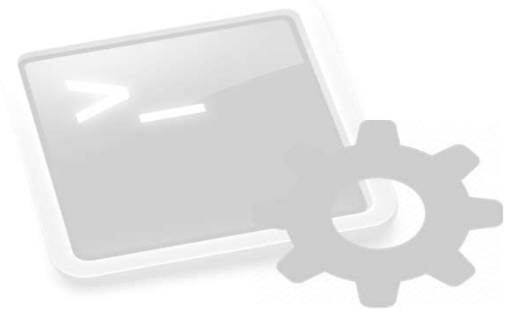
- Command **pvcreate**: creation of a Physical Volume.
  - Syntax: pvcreate [partition/whole disk/RAID device]
- Command **vgcreate**: creation of a Group Volume from multiple PVs.
  - Syntax: vgcreate [name-vol] [PVs list]
    - Example: vgcreate vg01 /dev/sdb /dev/sdc1 (group disk sdb and partition sdc1 in a GV).
- Command **lvcreate**: creation of a Logical Volume
  - Syntax: lvcreate [GV] –L[size] –n[name-vl]
    - Example: lvcreate vg01 –L1000M –nvol1 (after this we can create the FS with mkfs)
- Need more storage?
  - add a new Physical Volume to the Group Volume (**vgextend**)
  - Extend the Logical Volume to the larger Group Volume (**lvextend**)
  - Re-size the File System (resize2fs).
    - Can do this online !!! (…In contrast, reductions must be done offline)
  - We can also reduce VG and LV (**vgreduce**, **lvreduce**)

LVM Warnings

- Dual Boot
  - Windows does not support LVM; you will be unable to access any LVM partitions from Windows.
- Root FS (/) on LVM:
  - Not straightforward, ramdisk (initrd) must be updated properly.

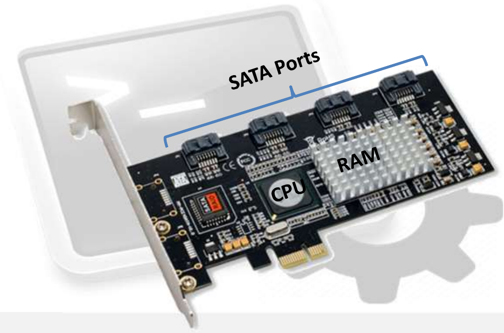
Index

- Introduction
- Adding a new Storage Device
- File System Consistency
- Managing Filesystem
- File System Security (disk encryption)
- **Advanced Device Management**
  - Logic Volume Management (LVM)
  - **Redundant Array of Inexpensive Disks (RAID)**
- Backup



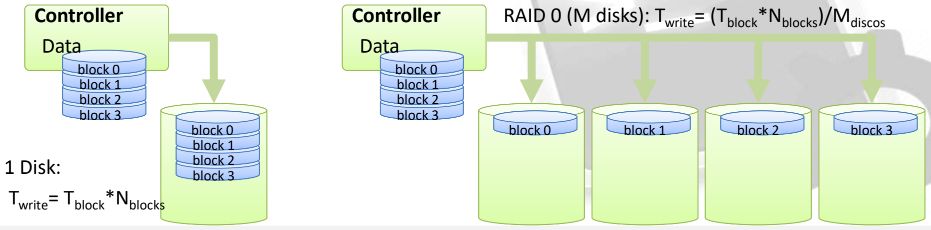
RAID (Redundant Array of Inexpensive Disks)

- Mechanism to provide reliability and performance in disks.
  - Make use of multiple disks to create the illusion of a disk with larger capacity, faster access and fault tolerant.
  - Transparent to the user and the OS (Hw RAID).
  - Different configuration options (Reliability vs Performance vs Capacity) denoted as levels (standard) [RAID0 … RAID6].
  - Can be implemented via HW or SW
    - HW Implementation: High efficiency but also high cost.
      - RAID Controller: CPU +dedicated sw, RAM + non-volatile memory.
    - SW Implementation: Efficient management of simplest RAID configs (0,1).



RAID 0 (Striping)

- Data are divided into segments (strips) and distributed among multiple disks.
  - parallel access to disks.
- **Performance**: improves read/write latency
  - Speed increases as the number of disks grows (also depends on data size).
- **Reliability**: no fault tolerance.
- **Capacity**: 100% storage utilized (no redundancy).



1 Disk:

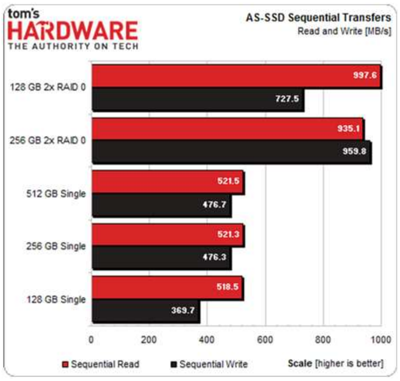
$T_{write} = T_{block} * N_{blocks}$

RAID 0 (M disks): $T_{write} = (T_{block} * N_{blocks}) / M_{discos}$

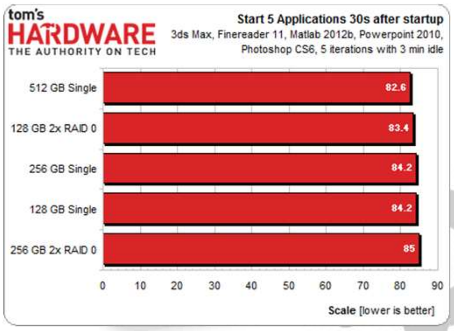
RAID 0 Performance

- One SSD Vs. Two in RAID: Which is better? (2013 analysis, a bit old)
  - https://www.tomshardware.com/reviews/ssd-raid-benchmark,3485.html

Sequential Read/Write Performance

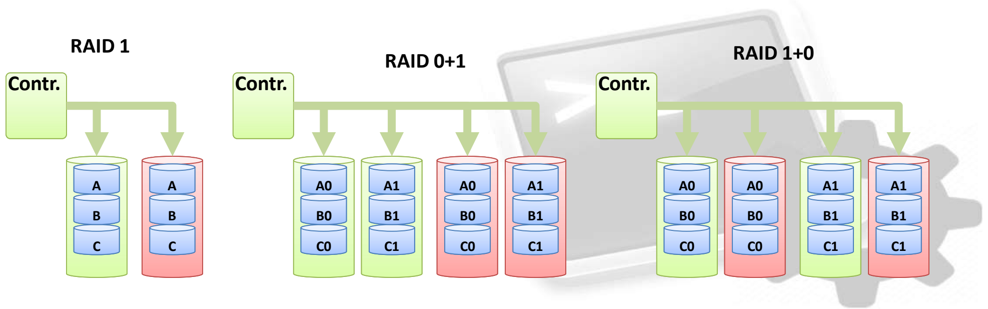


“Real World Benchmarks”



RAID 1 (mirroring)

- Employ a secondary disk to copy all data being modified
- **Performance**: benefits for reads, no improvement on writes (everything replicated)
- **Reliability**: High redundancy, one disk can fail.
- **Capacity**: 50% of total capacity available.



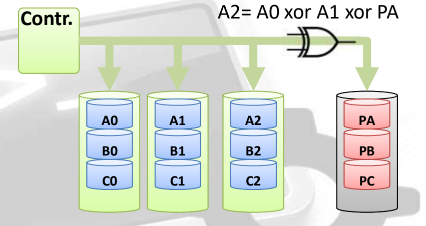
RAID 4 (Striping + parity)

- One disk stores information about the parity of the rest.
- Block-level division (1 strip= 1 block). Can access disks individually.
- **Performance**: High performance for reads. Bottleneck for writes.
- **Reliability**: Tolerance to 1 faulty disk.
- **Capacity**: Only 1 disk is not available.

PA= A0 xor A1 xor A2

If disk 2 fails:

A2= A0 xor A1 xor PA



¿How to calculate new parity after a write event?

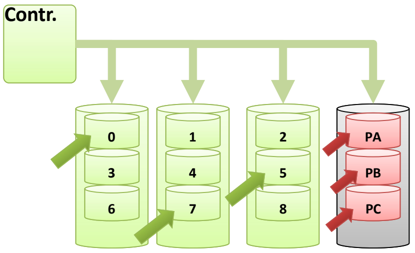
(Example: write in block B1)

Option1: Read the rest of blocks (B0, B2) and recalculate

Option2: Read the content of B1 and PB and calculate: $PB_{new} = PB_{old}$ xor $B1_{new}$ xor $B1_{old}$

Write problem in RAID 4

- Need to write in positions 0, 5, 7



Write process:

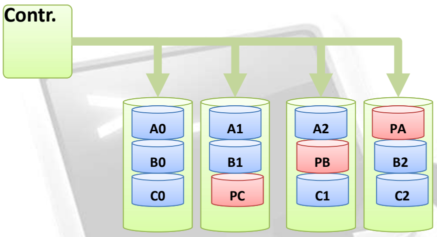
1. Read the blocks 0,5,7 and PA,PB,PC.
2. Calculate the new value of PA, PB, PC
3. Write new data blocks
4. Write new parity blocks

Serialized (writes in the same disk)-> Low performance

RAID 5(Striping+distributed parity)

- Parity information is distributed among all the disks.
- Similarly to RAID 4, block-level division (1 strip= 1 block).
- **Performance**: Eliminate the writes bottleneck.
- **Reliability**: Tolerates 1 faulty disk.
- **Capacity**: only 1 disk lost.

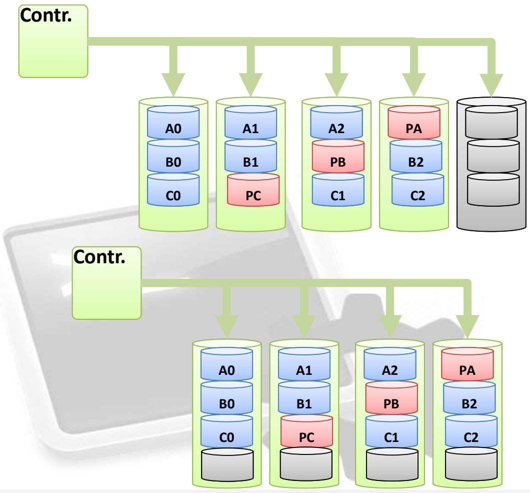
- RAID 6 (striping + double parity)
  - RAID 5 + double parity distribution
  - Tolerates two faulty disks.
- RAID 2, RAID 3
  - Parity control at a lower (than block) level.
  - Rarely employed.



RAID 5 + Spare Disk

- Add one additional disk to mitigate performance loss.
- Spare disks do not take part in the RAID until one active disk fails.
- On a device failure, reconstruction starts on the spare-disk.

- **RAID 5e** (striping + double parity)
  - Sparse capacity distributed among disks
  - No need for additional devices



RAID Administration (mdadm)

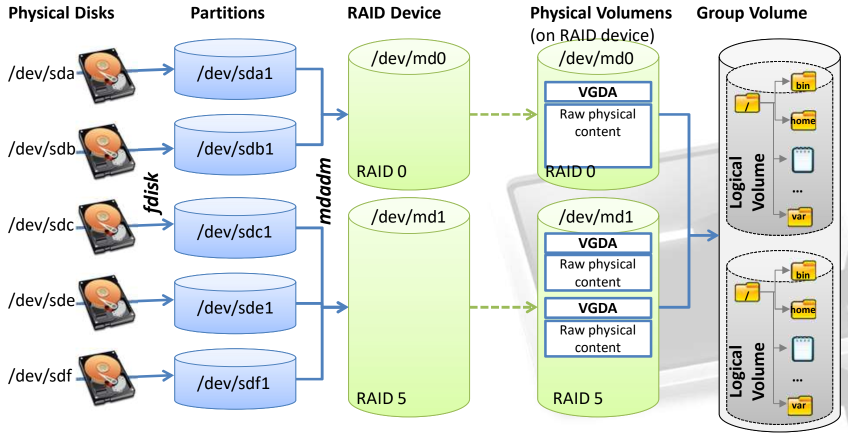
- **Creation** of a RAID device:
  - # mdadm --create /dev/md0 --verbose --level=0 --raid-devices=2 /dev/sdb /dev/sdc2
  - It is highly recommended the previous partitioning of disks (gdisk)
  - Creation process can be monitorized: # cat /proc/mdstat
  - Created a RAID in /dev/md0. On it we can create a File System (or a LVM Volume).
- **Monitorization** of RAID system:
  - # cat /proc/mdstat
  - # mdadm --monitor [options] /dev/md0
- **Elimination** (deactivation) of RAID:
  - “Stop” device: # mdadm --stop /dev/md0
  - Clean previous information from a RAID disk: # mdadm --zero-superblock /dev/sdX

RAID Administration (mdadm)

- Procedure for a **disk failure**:
  - Assume a RAID5 system, still operative with a significant performance degradation.
  - Broken disk can be automatically **restored**:
    1. Eliminate broken disk from RAID: # mdadm /dev/md0 -r /dev/sdc1
    2. Physically replace with another one (identical)
    3. Create the partitions as in the original: # gdisk /dev/sdc
    4. Add it to the RAID device: # mdadm /dev/md0 -a /dev/sdc1
    5. Monitorize the reconstruction process: # cat /proc/mdstat
  - We can simulate a disk failure:
    - # mdadm /dev/md0 –f /dev/sdc1
    - All the process log information in /var/log/messages
    - Faulty status is also displayed in /proc/mdstat

RAID + LVM

- RAID must be implemented below LVM



Physical Disks

Partitions

RAID Device

Physical Volumens (on RAID device)

Group Volume

/dev/sda

/dev/sdb

/dev/sdc

/dev/sde

/dev/sdf

fdisk

/dev/sda1

/dev/sdb1

/dev/sdc1

/dev/sde1

/dev/sdf1

mdadm

/dev/md0

RAID 0

/dev/md1

RAID 5

/dev/md0

VGDA

Raw physical content

RAID 0

/dev/md1

VGDA

Raw physical content

VGDA

Raw physical content

RAID 5

Logical Volume

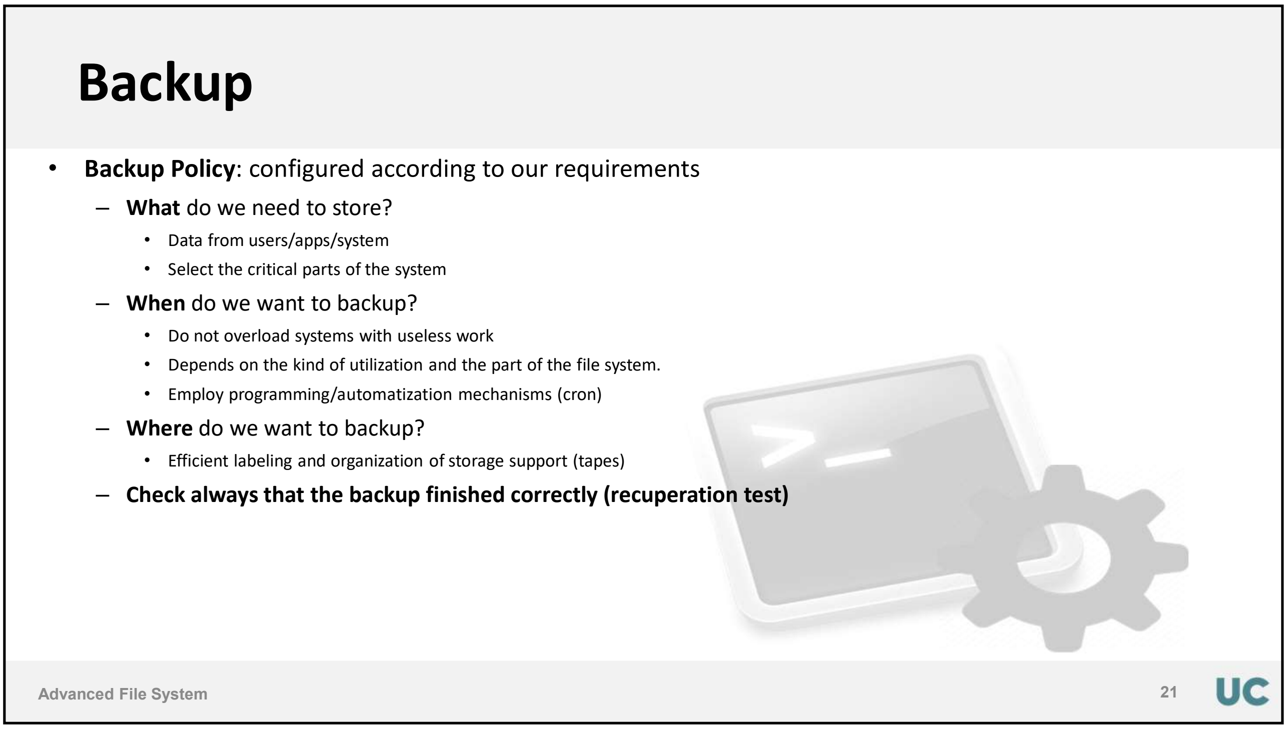
Logical Volume

Index

- Introduction
- The Linux Filesystem
- Adding a new Storage Device
- Managing Filesystem
- Advances Device Management
  - LVM
  - RAID
- **Backup**

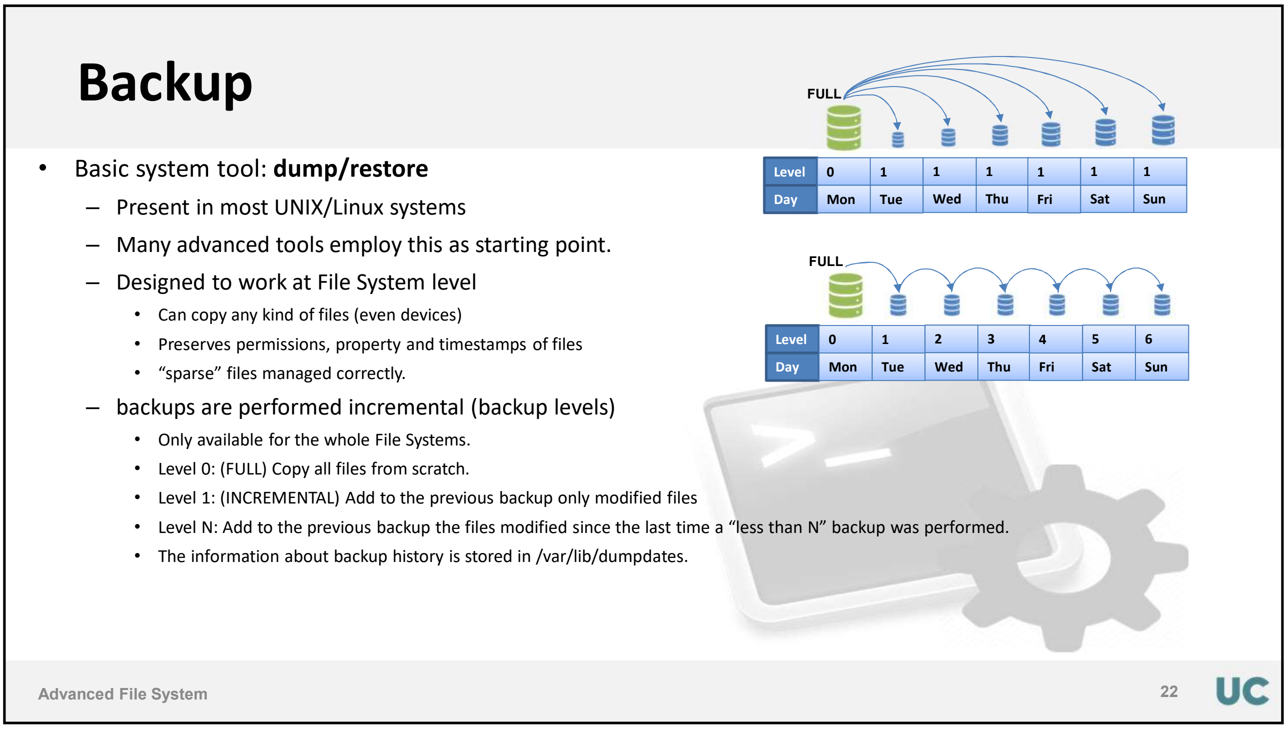
Backup

- RAID+journaling not enough to provide 100% availability.
- Essential: backup copies
  - Solution for multiple unexpected events, both HW and SW.
  - Mainly “the users”.
- Performed with dedicated resources:
  - Hard Disks
    - Exclusively dedicated to backup
    - NAS Servers
    - Disk hierarchy with decreasing performance
  - Tapes (or other magnetic support
    - LTO (Linear Tape-Open) (LTO-8 Ultrium):
      - 18TB capacity, 400MB/s transference.
    - Others: SAIT, AIT



Backup

- **Backup Policy**: configured according to our requirements
  - **What** do we need to store?
    - Data from users/apps/system
    - Select the critical parts of the system
  - **When** do we want to backup?
    - Do not overload systems with useless work
    - Depends on the kind of utilization and the part of the file system.
    - Employ programming/automatization mechanisms (cron)
  - **Where** do we want to backup?
    - Efficient labeling and organization of storage support (tapes)
  - **Check always that the backup finished correctly (recuperation test)**



Backup

- Basic system tool: **dump/restore**
  - Present in most UNIX/Linux systems
  - Many advanced tools employ this as starting point.
  - Designed to work at File System level
    - Can copy any kind of files (even devices)
    - Preserves permissions, property and timestamps of files
    - “sparse” files managed correctly.
  - backups are performed incremental (backup levels)
    - Only available for the whole File Systems.
    - Level 0: (FULL) Copy all files from scratch.
    - Level 1: (INCREMENTAL) Add to the previous backup only modified files
    - Level N: Add to the previous backup the files modified since the last time a “less than N” backup was performed.
    - The information about backup history is stored in /var/lib/dumpdates.

Level	0	1	1	1	1	1	1
Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun

Level	0	1	2	3	4	5	6
Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun

Backup

- Creation of backups with **dump** command
  - Syntax: dump -<level> <options> -f [destination] [File system]
    - Level: int from 0 (FULL) to 9
    - Option –f: destination of backup file. Can be a device file (tape)
    - Option –u: update the file /var/lib/dumpdates after the backup.
    - Example: # dump -0u -f /dev/tape /
- Recovery with **restore** command
  - restore –C: Compare the stored File system (from /)
  - restore –i: interactive operation with backup:
    - add/delete: files/dirs to the restoration list
    - cd/ls/pwd: move through the backup FS (Files with * are in the restoration list)
    - extract: restore the files from the list
  - restore –r: restore the whole file system
    - # restore –r –f <backup_file>
    - Executed inside the <destination> (preferably a brand-new mounted filesystem). Must be done level by level.