Disk subsytem

Managing disks, filesystem administration.

Disk drive installation

Connecting drive to computer

- Labeling and partitioning
- Creating logical volumes (LV)
- Creating filesystems on LV
- Setting up automatic mounting
- Setting swap file or swap partition

Hard disk types and interfaces

HDD - use magnet-coated platters **SSD** - use integrated circuits

Interfaces

SCSI from SCSI-1(1986, 5Mbps) undergone a lot of development over the years. Developed to connect all kind of periphery computer devices(HDD, printers, scanners,...) Not used in modern computers.

PATA was developed in 1986. It uses parallel transmission. It is also known as IDE. It uses an IDE cable to connect hard disks to the motherboard. It can transfer data up to 133 Mbps. This interface is also outdated.

SATA was developed in 2003. It uses serial transmission. It uses a serial cable to connect the hard disk to the motherboard. It is available in generations. The first generation SATA was launched as a 1.5 Gbps interface. The latest generation SATA is the third generation. It can transfer data up to 6 Gbps.

NVMe is a new interface type. Unlike SCSI, PATA, and SATA, it does not use a cable to connect the hard disk to the motherboard. It is available as a slot on the motherboard. It supports only SSD disks. It can transfer data up to 32 Gbps.

Other: SAS,FibreChannel,iSCSI,FCoE

BIOS/UEFI

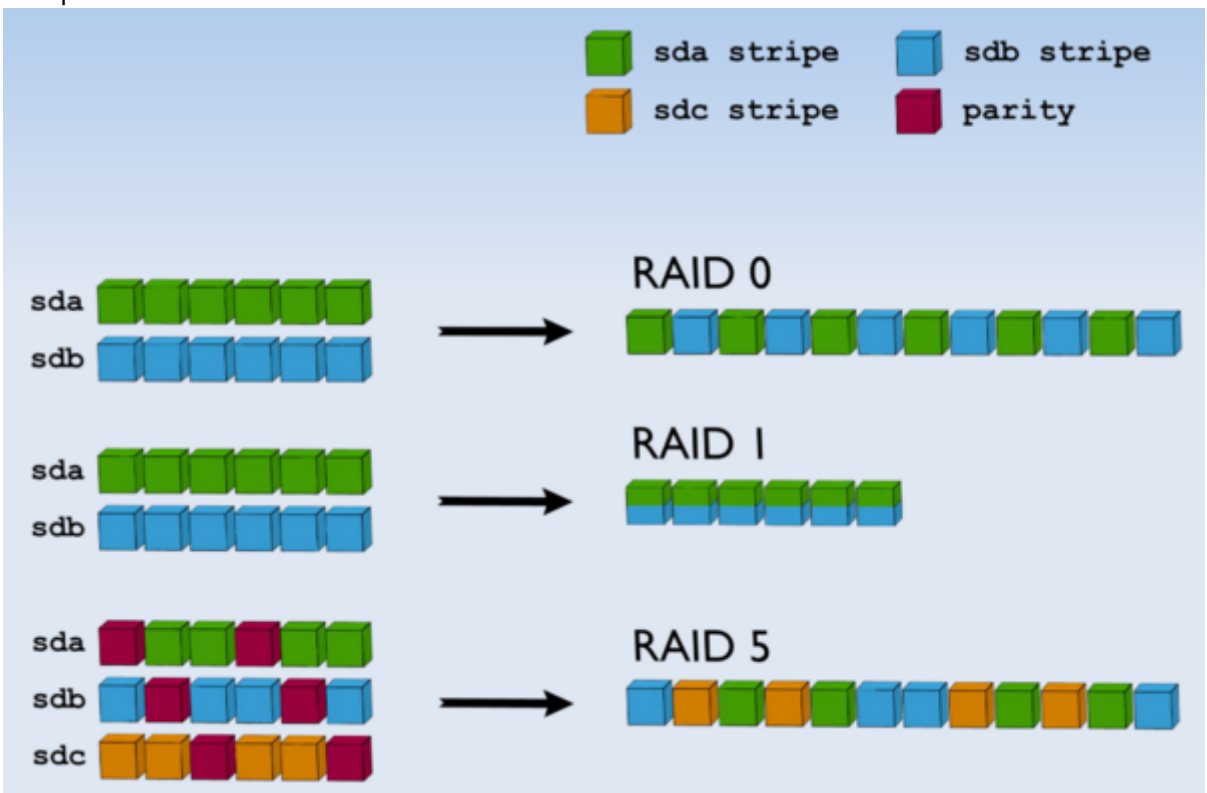
BIOS works in 16-bit mode. The 16-bit - recognizes and supports storage devices up to 2 TB.

UEFI (Unified Extensible Firmware Interface) is low-level software. It was developed to address the limitations of BIOS. It works similarly to BIOS but supports storage devices of all sizes. It works in 32-bit and 64-bit modes. It can boot from a disk larger than 2 TB.

MBR / GPT

MBR is a classical method - uses the first 512 bytes of the hard disk to save the partition and boot loader information. Of these 512 bytes, it uses only 64 bytes to store the partition table. Can store information about only 14 partitions.

GPT is a modern method. It uses 4Kb space to store partition information and boot loader program. Can store information about 128 partitions.

- ## RAID
  - RAID – Redundant Array of Inexpensive/Independent Disks
  - SW based RAID vs. Hardware RAID
  - Standard and widely used types
    - RAID 0, RAID 1, RAID 5, RAID 6
    - RAID 10 (1+0)
  - Not so common types
    - RAID 2, RAID3, RAID4, RAID7
    - RAID 0+1, RAID 50 (5+0), RAID 60 (6+0)
  - Spare devices
  - 

- ## Software RAID

`mdadm` – manage MD devices (Linux Software RAID)

  - ```
    mdadm --create raid_device --chunk=A --level=B --raid-devices=C --spare-devices=D devices
    ```
    ```

```
 #with disks
mdadm --create md0 -c 128 -l 1 -n 2 -x
0 /dev/sda /dev/sdb
```

```
#with partitions
mdadm --create md0 -c 128 -l 1 -n 2 -x
0 /dev/sda2 /dev/sdb2
```

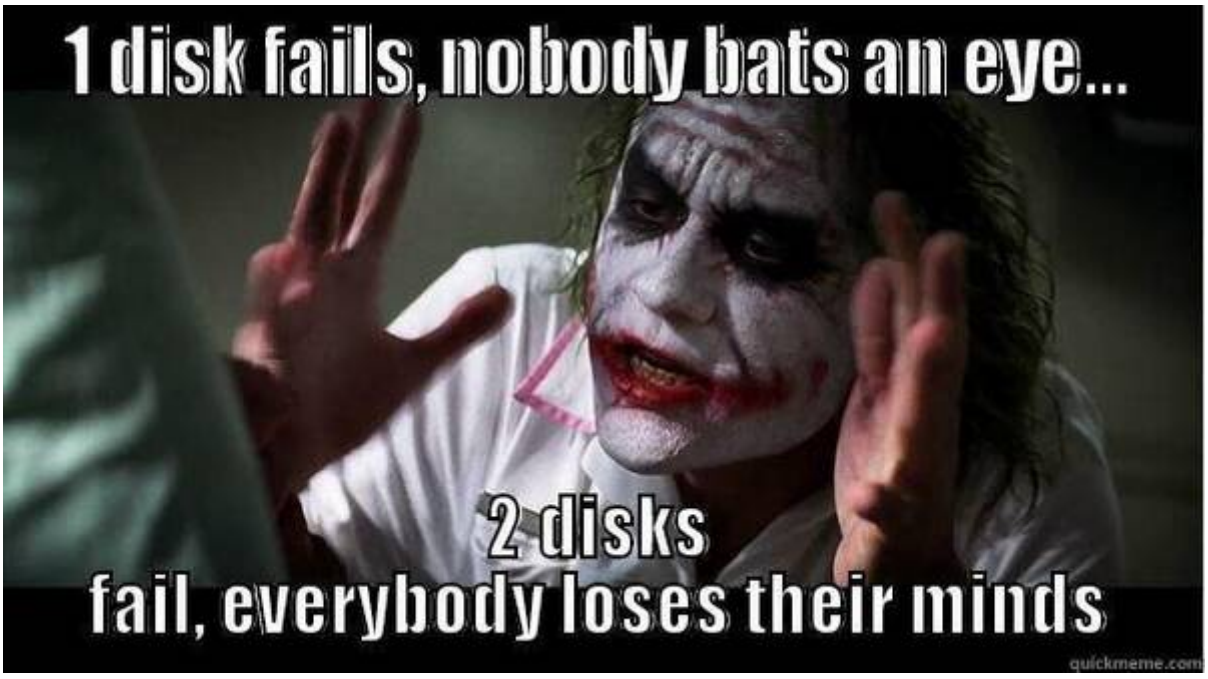
Manage software RAID

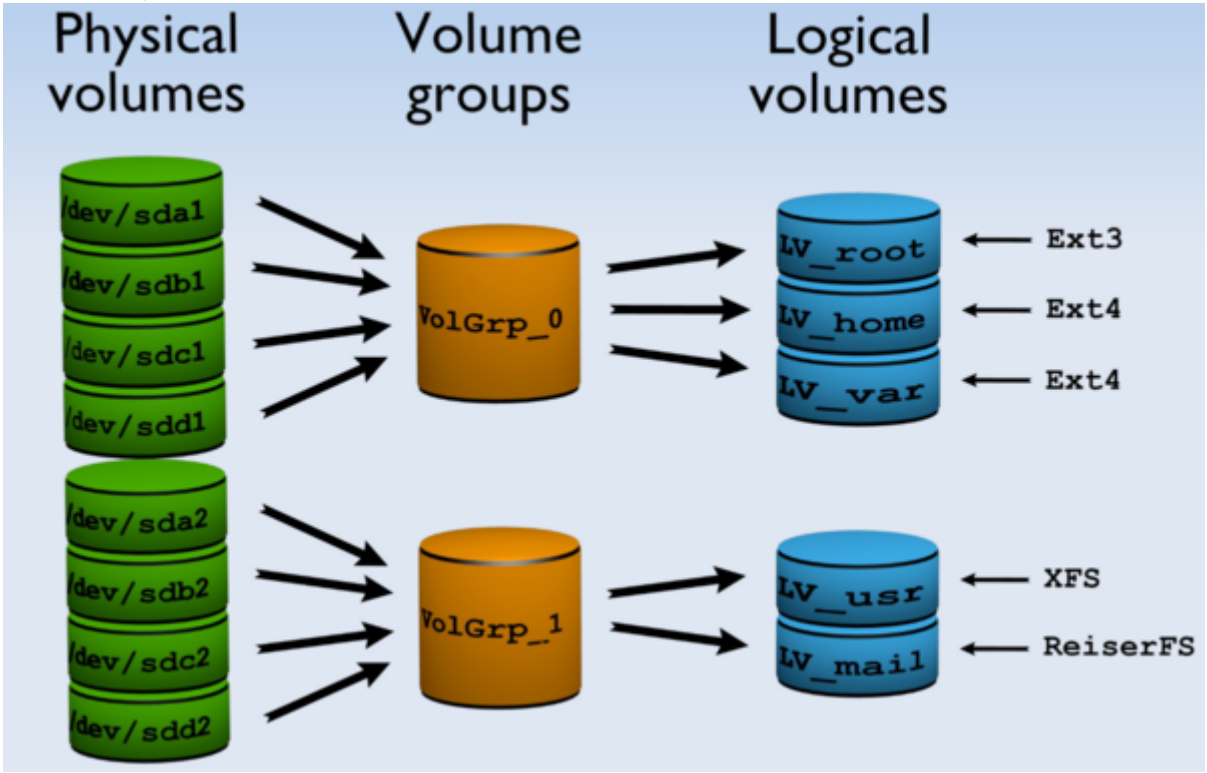
```
#add device to previously created RAID 1
mdadm --add /dev/md0 /dev/sdc
#mark device failed so the system stops using it
mdadm --manage --fail /dev/md0 /dev/sdc
#remove the device (so you can change the disk)
mdadm --remove /dev/md0 /dev/sdc
#...
#now replace the drive physically
#and go back to step one to add the new drive
```

- `/dev/md0` can be used all the time during the upper process as if nothing happened
- ofc for a true seamless operation the disk has to be hot-swap-able in all hardware/software and also physical manner

Use the right RAID level

- Selecting the proper RAID level for the job.



- # LVM
  - **LVM** – Logical Volume Manager (LVM1, LVM2)
    - Flexibility in allocating storage
    - Volumes can be resized and moved
    - Snapshots (can dramatically reduce the performance!)
  - **PV** - Physical volume
  - **PE** - Physical extends
  - **VG** - Volume groups
  - **LV** - Logical volumes
- 

- # LVM - usage
  - `apt-get install lvm2`
  - `pvcreate /dev/sdb1` – creates vg descriptor at begining of /dev/sdb1 partition
  - `pvdisplay` – check if the phys. volume is used by any logical volume
  - `vgcreate my_vol_grp /dev/sdb1 /dev/sdc1` create vol group
  - `vgchange -a y my_vol_grp` - activates a VG
  - `vgremove, vgextend, vgreduce`
  - `lvcreate -L20G -nMyLV_test my_vol_grp`
  - `lvcreate -L10G -s -n MySnapshot /dev/my_vol_grp/My_LV` - create snapshot, read more about snapshots.
- # File-systems

Disk file-systems

  - Unordered List ItemExt2, Ext3, Ext4, Btrfs, ZFS, JFS, XFS, NTFS, FAT,

FAT16, FAT32, ExFAT, GFS, UDF, etc.

Network filesystems

- NFS, ADF, SMB, (FTP, WebDAV)

Which filesystem is right for you?

- Journal support
- Access management (permissions)
- Extensibility, support by FS management utilities
- Special attributes support(chatr, selinux)
- Performance, availability, disaster recovery.
- Limits – storage, file size, number of files, etc.

Ext4 filesystem

- Ext(X) Default filesystem in many distributions
- Ext4 is successor of Ext3
- First stable release in 2008
- Storage limitation 1EiB (1024 PebiBytes) 1018 Bytes
- „Unlimited“ num. of files in one directory (32768 in Ext3)
- Total number of files in FS ~ 4.109 (232) (same as Ext3)
- Maximum file size of 16TiB (2TiB in Ext3)
- Inode – filesystem structure is based on inodes
  - Each object in the filesystem is represented by an inode
    - `ls -i /etc/passwd` - inode number for /etc/passwd

File-system utilities

- `mkfs, mkfs.ext3, mkfs.ext4,` etc. - create filesystem on device
  - `mkfs.ext3 /dev/sda2` – Creates Ext3 FS on /dev/sda2
- `tune2fs` – adjust parameters of Ext2,3,4 filesystems
  - `tune2fs -r 1000 /dev/sda2` - change number of reserved blocks
- `fsck, fsck.ext3`, etc – check and repair FS
  - `fsck.ext3 /dev/sda2` – check /dev/sda2 for errors
  - Repairing FS – it's strongly recommended to work on dismounted filesystem.

Mounting

- `mount / umount` – mount and dismount filesystem

```
mount -t ext3 /dev/sda2 /mnt/my_ext3_fs
umount /mnt/my_ext3_fs
umount /dev/sda2
```

- `/etc/fstab` – static info about the filesystem, see later
  - Automatic FS mounting after OS startup
  - mount points definitions

- # Labeling a partition
  - fdisk, parted, gparted(GUI)
    - `fdisk /dev/sdb`
  - mke2fs, tune2fs, e2label
    - `e2label /dev/sdb1 DiskName`
    - `tune2fs -L Name /dev/sdb1`
    - `mke2fs -L Name /dev/sdb1`
    - `mkfs.ext3 -L Name /dev/sdb1`

- # fstab

/etc/fstab file can be used to define how disk partitions, various other block devices, or remote file systems should be mounted into the file system.

```
#using kernel name descriptors
# <device>        <dir>        <type>        <options>        <dump>
<fsck>
/dev/sda1         /boot        vfat          defaults         0      2
/dev/sda2         /            ext4          defaults         0      1
/dev/sda3         /home        ext4          defaults         0      2
/dev/sda4         none         swap          defaults         0      0
#using labels
LABEL=Name /mnt/Name ext3 defaults 0 0
LABEL=HOME /home ext3 defaults 1 2
#File system UUIDs
UUID=CBB6-24F2                              /boot vfat   defaults  0
2
UUID=0a3407de-014b-458b-b5c1-848e92a327a3 /     ext4   defaults  0
1
UUID=b411dc99-f0a0-4c87-9e05-184977be8539 /home ext4   defaults  0
2
UUID=f9fe0b69-a280-415d-a03a-a32752370dee none  swap   defaults  0
0
```

- # fstab content
  - `device` describes the block special device or remote file system to be mounted.
  - `dir` describes the mount directory.
  - `type` the file system type
  - `options` associated mount options
  - `dump` is checked by the dump utility. This field is usually set to 0, which disables the check.
  - `fsck` sets the order for file system checks at boot time. For the root device it should be 1. For other partitions it should be 2, or 0 to disable checking.

- # fstab options
  - Ubuntu uses `relatime` as default for linux native file systems. This relates to when and

how often the last access time of the current version of a file is updated, i.e. when it was last read.
- `defaults` = `rw, suid, dev, exec, auto, nouser,` and `async`
- `ntfs/vfat` = permissions are set at the time of mounting the partition with `umask`, `dmask`, and `fmask` and can not be changed with commands such as `chown` or `chmod`. It is advised `dmask=027,fmask=137` (using `umask=000` will cause all your files to be executable). More permissive options would be `dmask=000,fmask=111`.
- For mounting samba shares you can specify a `username` and `password`, or better a credentials file. The credentials file should be owned by root.root with permissions = 0400 .

fstab options continue

- `sync/async` - All I/O to the file system should be done (a)synchronously.
- `auto` - The filesystem can be mounted automatically (at bootup, or when mount is passed the -a option). This is really unnecessary as this is the default action of mount -a anyway.
- `noauto` - The filesystem will NOT be automatically mounted at startup, or when mount passed -a. You must explicitly mount the filesystem.
- `dev/nodev` - Interpret/Do not interpret character or block special devices on the file system.
- `exec/noexec` - Permit/Prevent the execution of binaries from the filesystem.
- `suid/nosuid` - Permit/Block the operation of `suid`, and `sgid` bits.
- `ro` - Mount read-only. `rw` - Mount read-write.
- `user` - Permit any user to mount the filesystem. This automatically implies `noexec, nosuid,nodev` unless overridden.
- `nouser` - Only permit root to mount the filesystem. This is also a default setting.
- `defaults` - Use default settings. Equivalent to `rw, suid, dev, exec, auto, nouser, async`.
- `netdev` - this is a network device, mount it after bringing up the network. Only valid with fstype nfs.

Warning

Kernel name descriptors for block devices are not persistent and can change each boot, they should not be used in configuration files (including /etc/fstab).

```
#to check your UUIDs or LABELs use:
blkid # locate/print block device attributes
lsblk -f # list block devices
```

Ramdisk

- A ramdisk is a **volatile storage space defined in the RAM memory**. Using this feature increases file processing performance ten times over the best SSD hard disks. Implementing a ramdisk is very advantageous for users whose tasks require significant amounts of hardware resources.
- A ramdisk is a volatile space, all information stored in it will be lost if the device is turned off or reboots.
- Yes, ramdisk consumes some of your RAM space so don't get too wild with the size

- Ramdisk is a good option for frequent read/write operations, sometimes even a must (ie. SD-cards as disk in embedded systems)

```
mkdir -p /media/somedir

mount -t tmpfs -o size=2048M tmpfs /media/somedir/
```

Disk usage

check the usage of the disk, free space, who consumes the space etc

Native commands

- df - report file system disk space usage

```
user@machine:/tmp$ df ./
Filesystem     1K-blocks  Used Available Use% Mounted on
tmpfs            6451972 32472   6419500   1% /tmp
user@machine:/tmp$
```

- du - estimate file space usage

```
user@machine:/tmp$ du -ah ./pokus/
3.0M    ./pokus/file1.txt
4.0K    ./pokus/file.txt
3.1M    ./pokus/
user@machine:/tmp$ du -hs ./pokus/
3.1M    ./pokus/
user@machine:/tmp$
```

Dedicated utilities

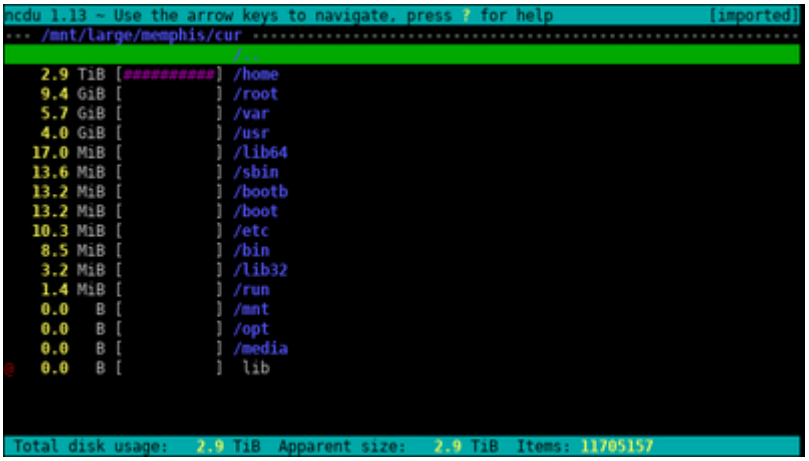
NOTE: may require to be installed, ie not being installed by default on your distro

- duc - index, query and graph disk usage

```
user@machine:/tmp$ duc index /usr #first make index
user@machine:/tmp$ duc ls -Fg /usr/ #now see the statistics within the
indexed dir
  6.9G lib/
[++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++]
  3.5G share/
[++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++]
555.5M local/   [++++++++++++]
425.3M bin/     [+++++++++]
317.8M include/ [+++++++]
291.0M src/     [++++++]
 25.9M sbin/    []
 15.3M libexec/ []
```

```
 14.7M games/    []
user@machine:/tmp$
```

- ncdu - NCurses Disk Usage



Mounting /tmp/ as ramdisk

- Mounting `/tmp` to the ramdisk
  - may reduce the SSD disk wearing off
  - may make programs that use /tmp faster
  - ofc consumes some RAM, so you rather use it only if there is enough
  - beware of `noexec` option, this adds some security, but some utilities may fail with it
    - `mount -o remount,exec /tmp` to remount it temporarily with executable flag
- To mount /tmp as ramdisk at boot add the following line in to your /etc/fstab
  - Do not forget to change the size to your preferences with respect to your actual RAM size

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,nodev,noexec,mode=1777,size=1G
0 0
```

/dev/shm ramisk by default

The `/dev/shm` file system in Linux is a temporary filesystem (a tmpfs) that stores files in memory (RAM), allowing for fast read and write access. It is typically used for inter-process communication (IPC) via shared memory in Linux systems, but can also be utilized for any other temporary storage needs.

One might want to add the size limitations to the shared memory filesystem, add the following line to the `/etc/fstab`... tweak the `size` to your preferences ofc.

```
tmpfs /dev/shm tmpfs defaults,size=2G 0 0
```

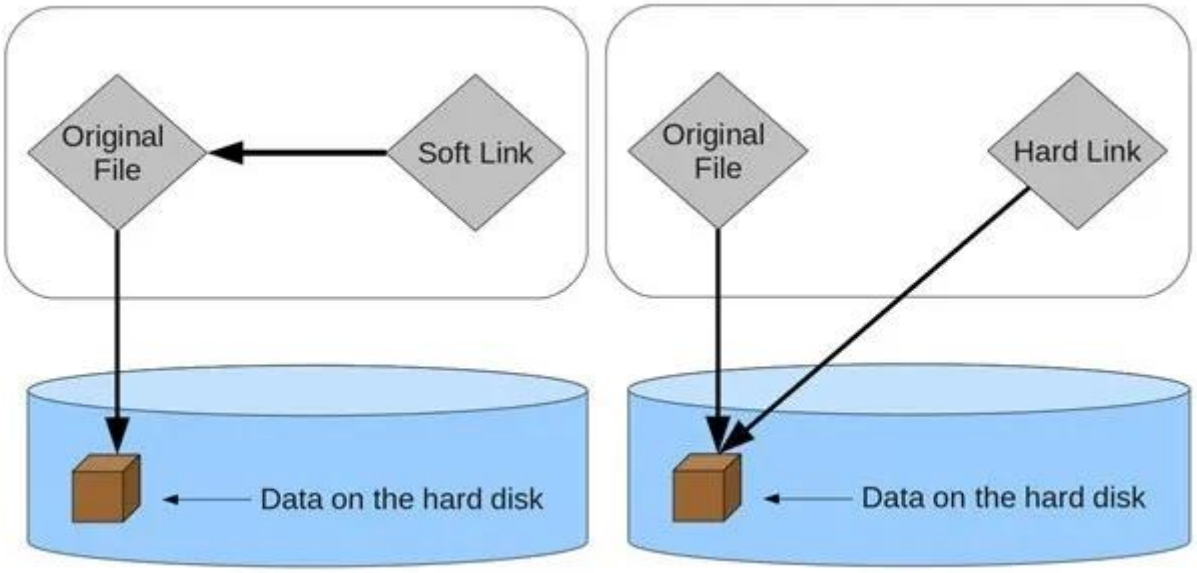
- # Links

HardLink

- Only files
- In context of one filesystem and LV
- `ln /etc/rc.local ~/h_rc.local`

Symbolic Link (symlink, softlink)

- Files & Directories, across other filesystems or volumes
- `ln -s /etc/rc.local ~/s_rc.local`



- # Swap file / partition

- Swap partition
  - `mkswap /dev/sda4` - format partition to be used as swap
  - `swapon /dev/sda4` - turn on the swap
  - `swapoff /dev/sda4` - turn it off
- Swapfile
  - `dd if=/dev/zero of=/swap.img bs=512k count=128`
    - or `fallocate -l 1G /swap.img`
    - change its permissions
      - `sudo chmod 600 /swapfile`
  - `mkswap /swap.img`
  - `swapon /swap.img`
  - you might also want to add it to your fstab to make it permanent
    - `echo '/swap.img none swap sw 0 0' | sudo tee -a /etc/fstab`
- Sparse files
  - `dd if=/dev/zero of=/swap.img bs=512k count=1 seek=128`

- # Encrypted storage

- **Full Disk Encryption** - Full disk or partition encryption is one of the best ways of protecting your data. Not only is each file protected but also the temporary storage that

may contain parts of these files is also protected. Full disk encryption will protect all of your files so you do not have to worry about selecting what you want to protect.
  - **File-Based Encryption** - File-based encryption is used to protect the contents of files on mobile storage devices, such as CDs, flash drives, or external hard drives. Some file-based encryption solutions may leave remnants of the encrypted files that an attacker who has physical access to your computer can recover under some circumstances. To protect the contents of these files from attackers who may have access to your computer, use file-based encryption combined with another solution, such as full disk encryption.

- # Full Disk Encryption

  - **DM-Crypt** - A Linux kernel-level encryption tool. An operating system uses mounting to access a file system by attaching it to a directory (mount point).
  - **VeraCrypt** - on-the-fly encryption (OTFE) for free. With the software, you can create a virtual encrypted disk within a file that works just like an actual disk. As well as encrypting a partition, it can authenticate the entire storage device before booting.
  - **Cryptmount** - allows users to access encrypted filesystems without having root privileges on GNU/Linux systems. You will need Linux 2.6 or higher to run it. The program is capable of handling both encrypted partitions and encrypted files.
  - **eCryptfs** - Enterprise Cryptographic Filesystem (eCryptfs) allows encrypting disks on Linux. Unlike dm-crypt, which offers a block device encryption layer, eCryptfs provides an actual stacked cryptographic file system.
  - **loop-AES** -encrypts both file systems and swap partitions quickly and transparently. Loop-AES is suitable for encrypting not only disk partitions but swap space, removable media, and other devices. Various measures are available to strengthen encryption, including passphrase seeds, multiple hash iterations, MD5 IV, and alternating encryption keys.

- # dm-crypt

  - `apt-get install cryptsetup`
  - `cryptsetup –verify-passphrase –verbose luksFormat /dev/sdd1`
  - `cryptsetup luksOpen /dev/sdd1 mydisk`
  - `mkfs.ext3 /dev/mapper/mydisk`
  - `mkdir /mnt/mydisk`
  - `mount /dev/mapper/mydisk /mnt/mydisk`
  - `umount /mnt/mydisk`
  - `cryptsetup luckClose /dev/mapper/mydisk`

- # encfs

```
#install the package
sudo apt install encfs
#create encrypted directory and mount it to visible one
encfs ~/.encrypted ~/visible
#close the visible dir
fusermount -u ~/visible
#later mount it again
encfs ~/.encrypted ~/visible
#etc...
```

- # eCryptfs

  eCryptfs is a POSIX-compliant enterprise cryptographic stacked filesystem for Linux. eCryptfs stores cryptographic metadata in the header of each file, so that encrypted files can be copied between hosts; the file will be decrypted with the proper key in the Linux kernel keyring. There is no need to keep track of any additional information aside from what is already in the encrypted file itself. You may think of eCryptfs as a sort of “gnupg as a filesystem”.

  Read more about how to use it.

- ## Exercises

  - Create `test_1.txt` file in your home directory
  - Create hardlink `test_2.txt` (link to test_1.txt)
  - Create symlink `test_3.txt` (link to test_1.txt)
  - Add text message into `test_1.txt` file
  - Check with `ls -l –color`
  - Check content of `test_2.txt` and `test_3.txt`
  - Rename test_1.txt to test.txt
  - Check again with `ls -l –color`
  - Check content of `test_2.txt` and `test_3.txt`
  - Use dd utility to increase swap space (make 64MB swapfile)
  - Create 3 x 64MB drives in virtualbox
    - Make RAID 0 device on /dev/sdb and /sdc
    - Use LVM to create 32MB LV on sdd drive
    - Use LVM to create snapshot of previously created LV