ITINERIS

Unix kernel and its storage subsystem

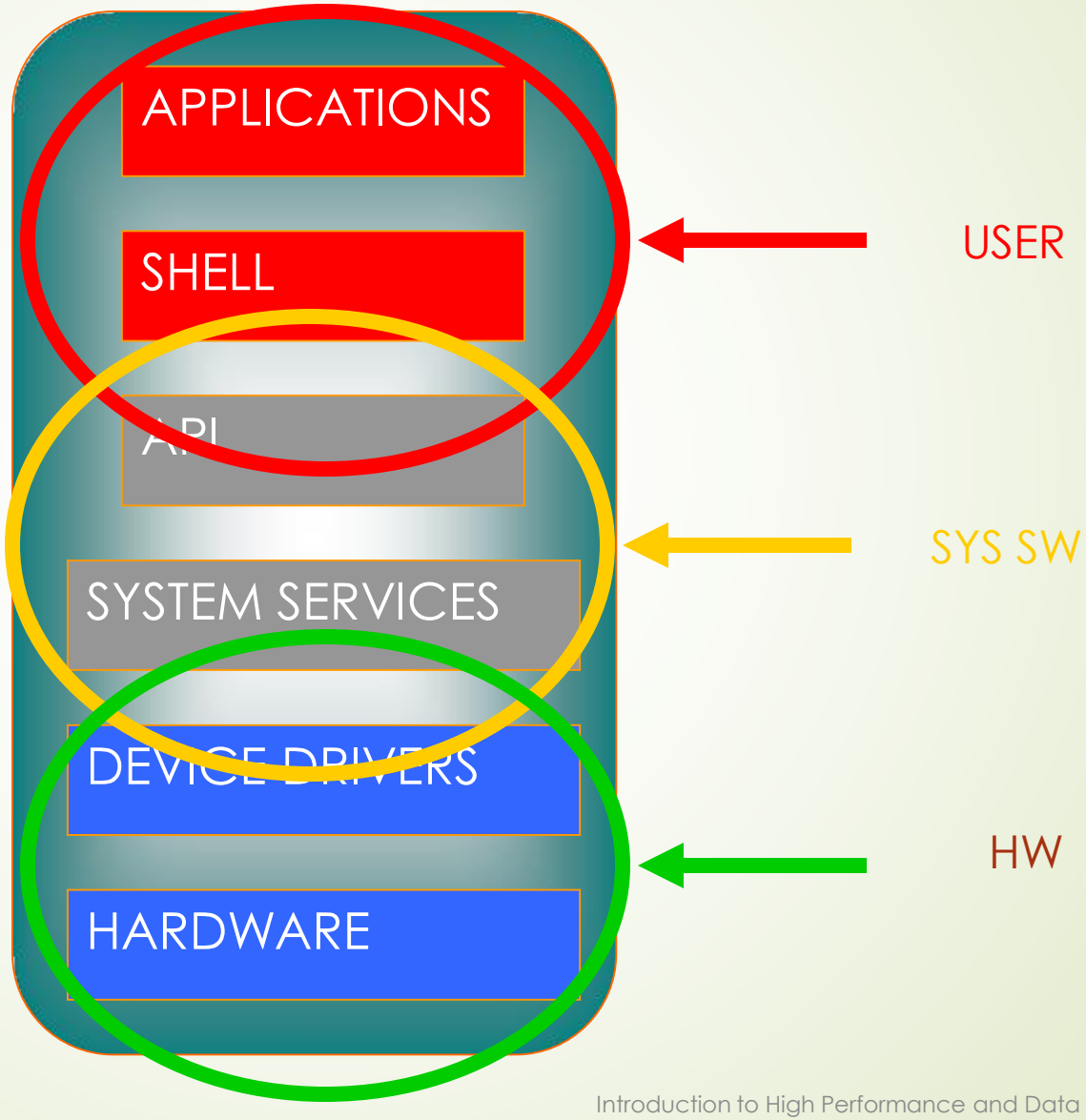
A brief introduction to the core of a Linux system

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

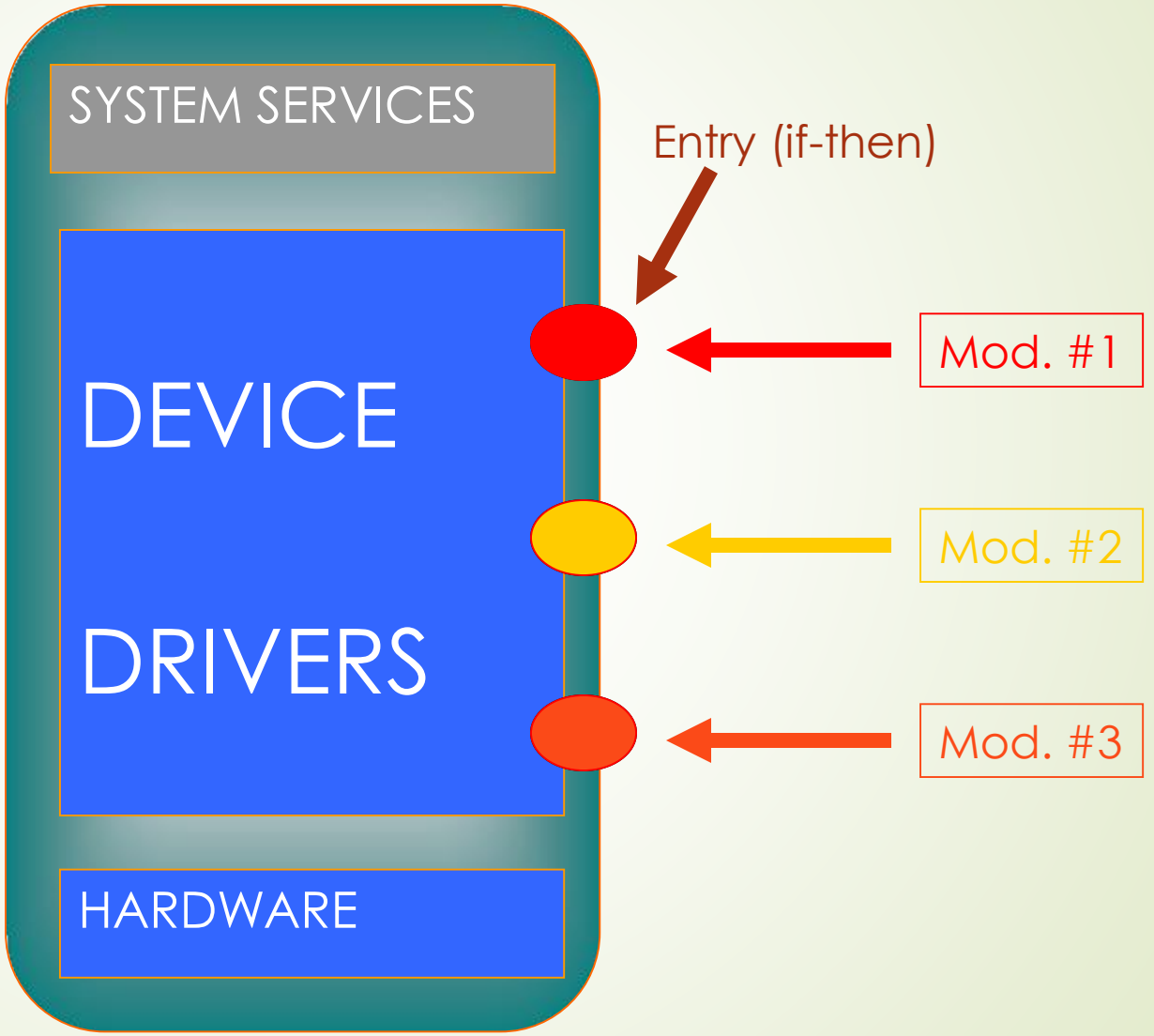
The structure of Unix Kernel

A layered structure



Kernel Device Drivers

A modular structure (module [*.ko])



Kernel Device Drivers

At boot time the kernel makes a check

```
if (module exist) then load it
```

or at compilation time (when you build your kernel image) the kernel has to have defined all the ***entries*** for each hardware device present on the system.

It is a duty of the hardware producer (and/or of the Linux OS community) to provide the *modules* (***device drivers***) for having those hardware working properly on your machine.

The case of Nouveau Device Drivers

To prevent conflicts and ensure proper NVIDIA driver loading, it's crucial to blacklist the `nouveau` driver before installing CUDA. This involves modifying system configuration files to prevent `nouveau` from loading at boot time. Specifically, this can be achieved by creating a file in `/etc/modprobe.d/` that instructs the system to ignore `nouveau`. Here's a step-by-step guide:

Create a Blacklist Configuration File:

- Open a terminal.
- Create a new file with a name like `nvidia-blacklist.conf` in the `/etc/modprobe.d/` directory:

```
sudo nano /etc/modprobe.d/nvidia-blacklist.conf
```

- Add the following lines to the file:

```
blacklist nouveau
options nouveau modeset=0
```

Save and close the file.

Update Initramfs (Optional, but Recommended):

- Some distributions require updating the initramfs to ensure the blacklist is applied during the boot process.
  - Use the following command:

```
sudo update-initramfs -u
```

Update GRUB (If Needed):

- If you're using GRUB, you can also add a kernel parameter to blacklist `nouveau`.
  - Edit `/etc/default/grub`:
- Add `nouveau.blacklist=1` to the `GRUB_CMDLINE_LINUX_DEFAULT` line (e.g., `GRUB_CMDLINE_LINUX_DEFAULT="quiet nouveau.blacklist=1"`).
- Save and close the file, then update GRUB

```
sudo update-grub
```

Reboot:

- After making these changes, reboot your system to ensure the blacklist is applied.
- After rebooting, you can check if the `nouveau` driver is loaded by running:

```
lsmod | grep nouveau
```

If no output is shown, the `nouveau` driver is successfully blacklisted.

Kernel Subsystems

The modular structure of the POSIX kernel permits to manage the hierarchy of modules with a schema where some of them are organized as a set. This self-consistent set of modules is called (kernel) **subsystem**.

In this way we can operate on a group of module as a whole thus facilitating the management of the undelying hardware at which they refer to.

So we could speak of I/O subsystem, interconnect (or communication) subsystem, memory subsystem, and so on.

Kernel Subsystems

As an examples, let's have a look at the **IPC** (Inter Process Communication) subsystem which includes the management of the kernel queues (que) the share memory segments (shm) and semaphores (sem). All of these modules may be managed through a common set of parameters.

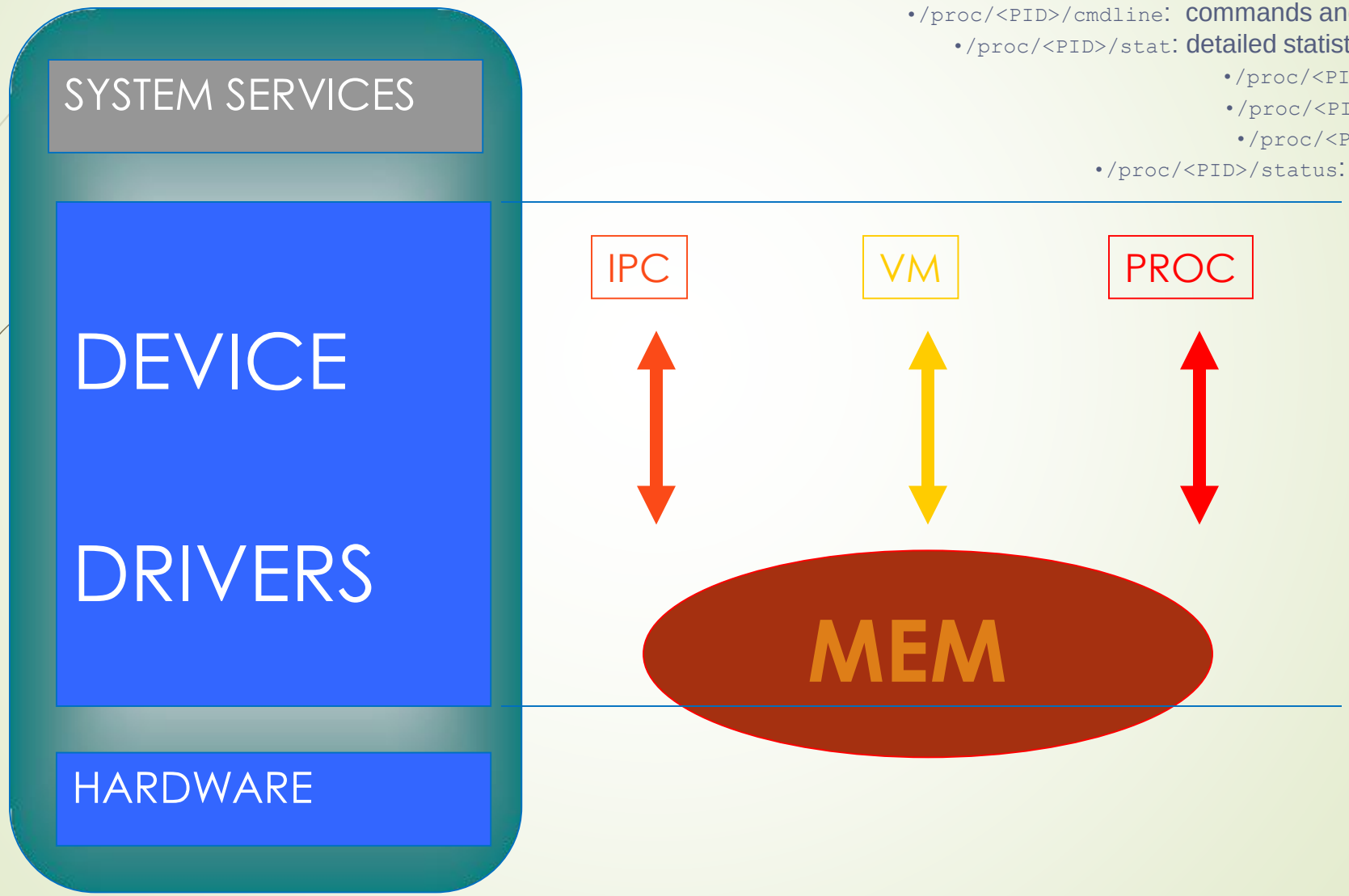
IPC subsystem control the communication among processes and it is of vital importance in any Unix OS.

The IPC subsystem interact with the (**VM/mem/PROC**) subsystems from which it get the reference to the memory pages addressable in kernel space and process info.

Kernel Subsystems

`procfs:`

- `/proc/<PID>/cmdline`: commands and arguments for execution.
- `/proc/<PID>/stat`: detailed statistic information per process.
- `/proc/<PID>/environ`: env variables.
- `/proc/<PID>/mem`: allocated memory.
- `/proc/<PID>/cwd`: Current directory.
- `/proc/<PID>/status`: Status of the PID process.



Kernel Subsystems

Among the various subsystem of a Unix OS, one is particularly useful for us: the **Unix filesystem**.

We can say, aproximately, that a file system is the transposition of the disk geometry into the kernel logic for the management of *block-devices*.

But what are those block-devices we are speaking about…block-devices provide buffered access to hardware devices, and provide some abstraction from their specifics; see /dev

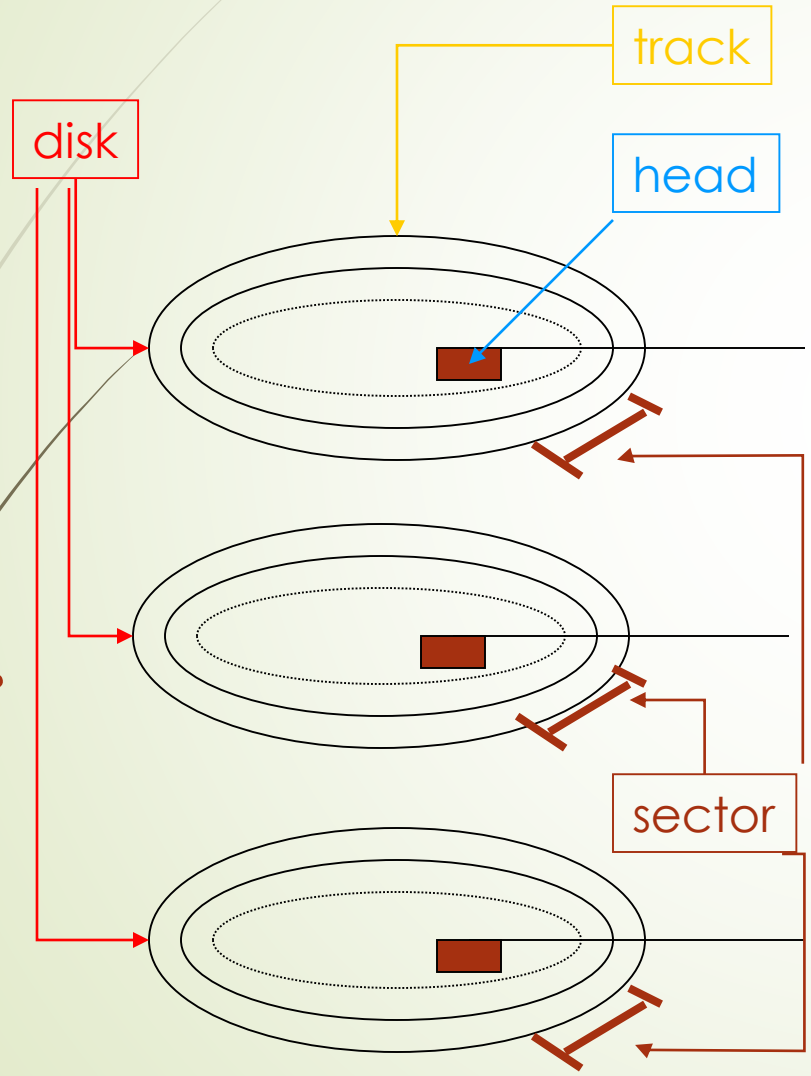
Kernel Subsystems

One of the first and mostly used filesystems for disk drives avalable for Unix kernels is called Unix File System (UFS) and has its own standard known as BSD4.2.

In UFS there exist a direct correlation between the elements of the hard disk geometry (disks, tracks, sectors, heads, cylinders, etc.) and their representation at higher level into the logic of kernel, that is, the i-node.

An **inode** (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data.

Kernel Subsystems



I-node

:=

F[t1,tr1,s1(tr1),

t2,tr2,s2(tr2),

…

tN,trN,sN(trN)]

:=

F[cyl/1-N]

Kernel Subsystems

Structure of a modern filesystem

At present a filesystem implements two basic concepts: the fileset (**LV – Logical Volume**) and il filedomain (**Volume Group – VG**).

Fileset and filedomain thus enable a **2 levels** structure of filesystems which *decouples* the layer of the physical storage from the hierarchy of the directory tree.

Kernel Subsystems

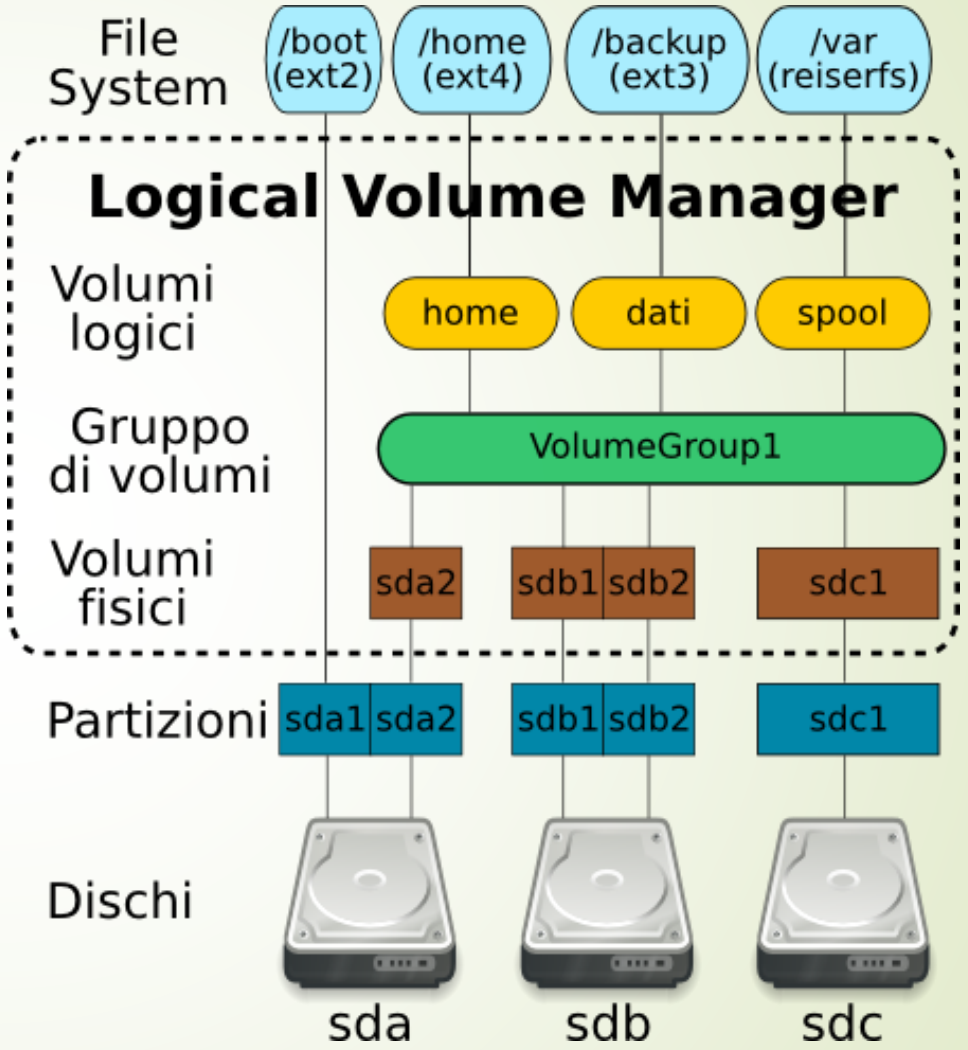
A fileset has a logic structure resembling a traditional UFS made out of a hierarchy of directory and file names of which one typically «**mount**».

Modern filesystems go beyond the limit of traditional UFS and permit to share a common set of storage elements call **filedomain**.

Un filedomain (VG) is then, a set of storage which can be managed indipendently from the directory structure.

In this way, you sysadmin may add or remove ***volumes*** (partitions or filesets) without altering the directory structure.

LVM & Linux


File System
/boot (ext2)
/home (ext4)
/backup (ext3)
/var (reiserfs)
Logical Volume Manager
Volumi logici
home
dati
spool
Gruppo di volumi
VolumeGroup1
Volumi fisici
sda2
sdb1
sdb2
sdc1
Partizioni
sda1
sda2
sdb1
sdb2
sdc1
Dischi
sda
sdb
sdc

LVM & Linux

A practical example:

Let us suppose to have to partitions: `/dev/sda1` e `/dev/sdb1`.

1.PV:

- `sudo pvcreate /dev/sda1`
- `sudo pvcreate /dev/sdb1`

2.VG:

- `sudo vgcreate myvg /dev/sda1 /dev/sdb1`

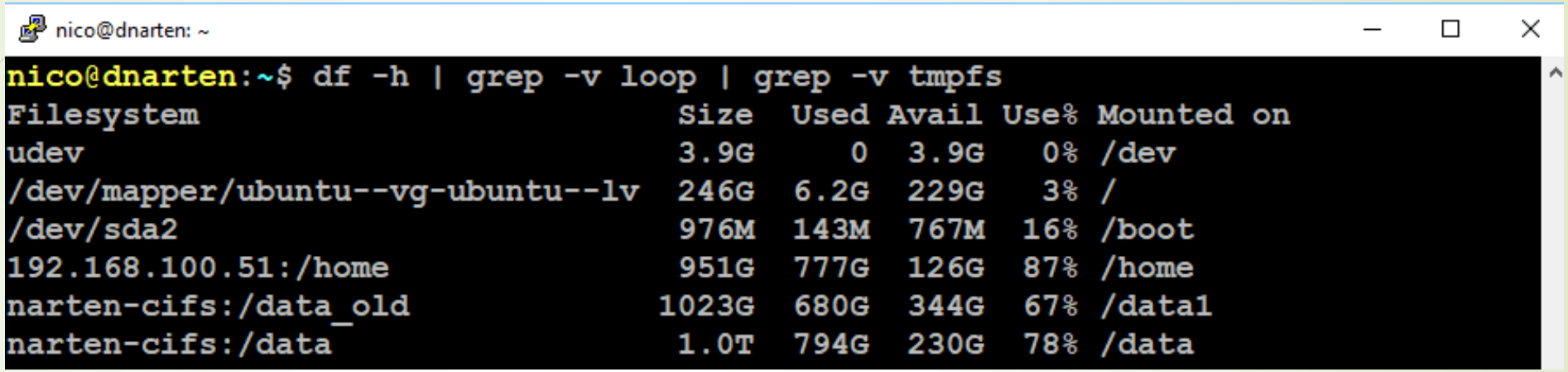
3.LV:

- `sudo lvcreate -L 10G myvg home` (crea un LV di 10 GB chiamato "home")
- `sudo lvcreate -L 20G myvg data` (crea un LV di 20 GB chiamato "data")

4.Mount points:

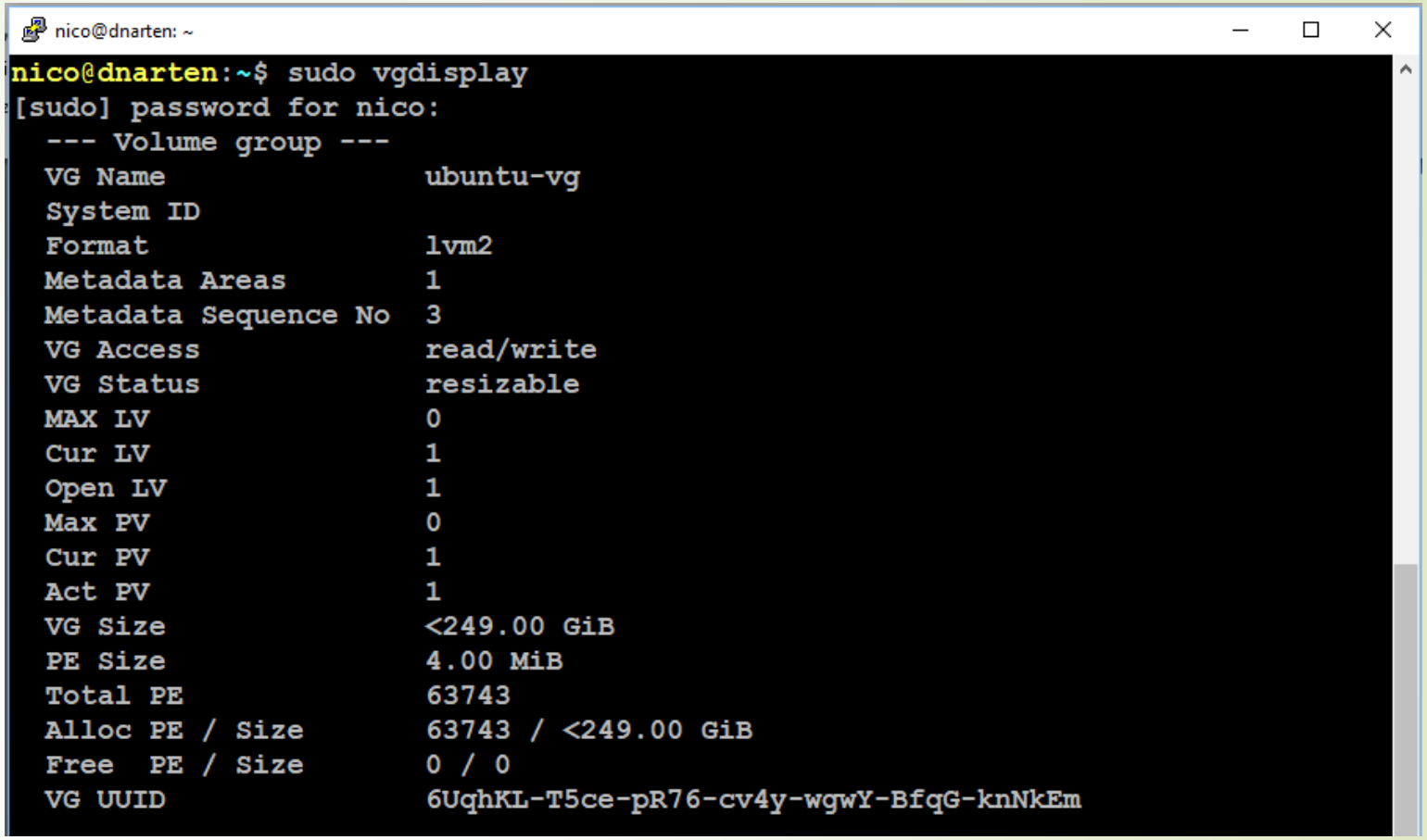
- `sudo mkdir /mnt/home`
- `sudo mkdir /mnt/data`
- `sudo mount /dev/myvg/home /mnt/home`
- `sudo mount /dev/myvg/data /mnt/data`

See also /etc/fstab file and fdisk utility from command line

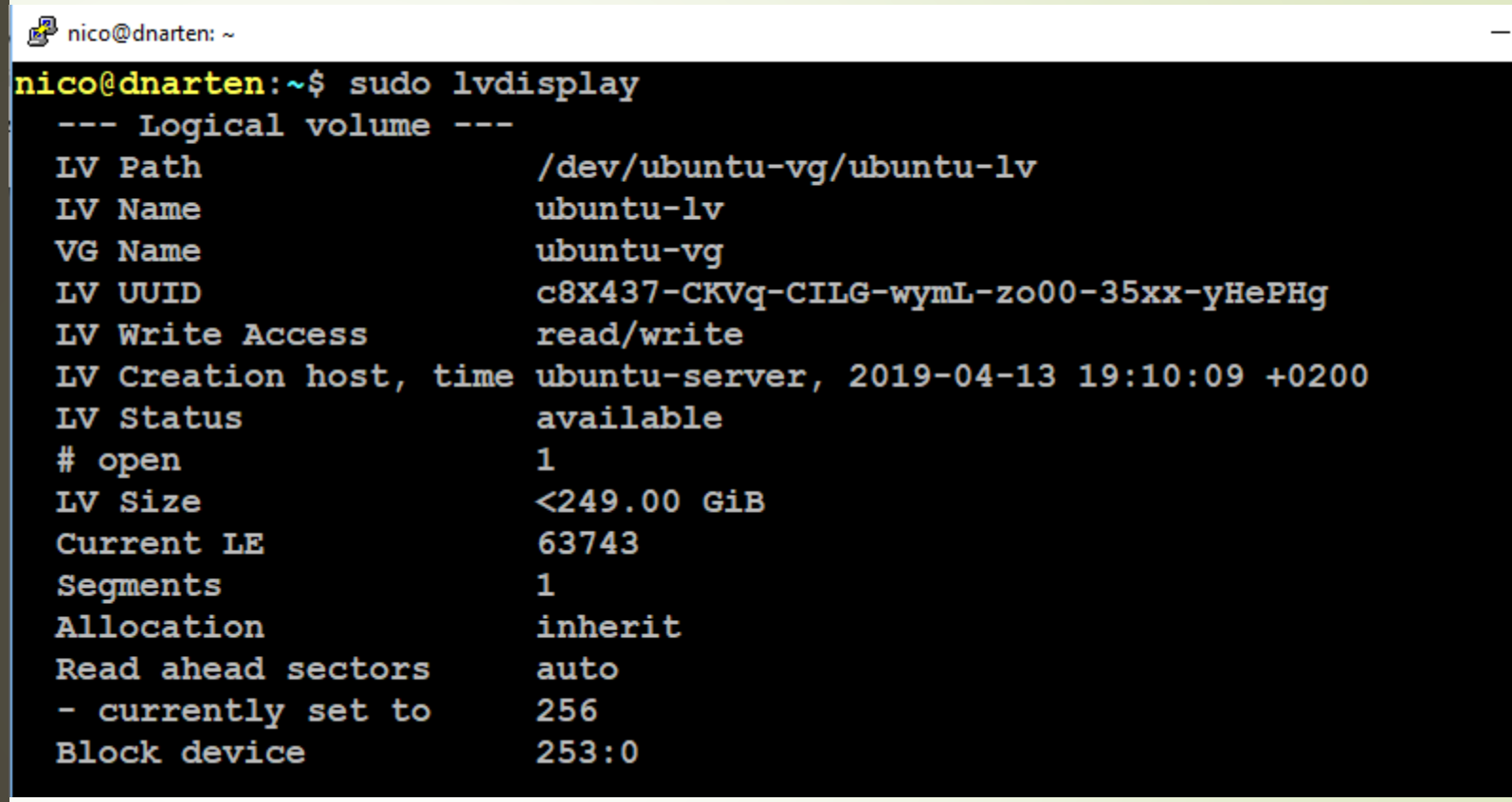

nico@dnarten: ~
nico@dnarten:~$ df -h | grep -v loop | grep -v tmpfs
Filesystem Size Used Avail Use% Mounted on
udev 3.9G 0 3.9G 0% /dev
/dev/mapper/ubuntu--vg-ubuntu--lv 246G 6.2G 229G 3% /
/dev/sda2 976M 143M 767M 16% /boot
192.168.100.51:/home 951G 777G 126G 87% /home
narten-cifs:/data_old 1023G 680G 344G 67% /data1
narten-cifs:/data 1.0T 794G 230G 78% /data

LVM & Linux

LVM & Linux


nico@dnarten: ~
nico@dnarten:~$ sudo vgdisplay
[sudo] password for nico:
--- Volume group ---
VG Name ubuntu-vg
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 3
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 1
Open LV 1
Max PV 0
Cur PV 1
Act PV 1
VG Size <249.00 GiB
PE Size 4.00 MiB
Total PE 63743
Alloc PE / Size 63743 / <249.00 GiB
Free PE / Size 0 / 0
VG UUID 6UqhKL-T5ce-pR76-cv4y-wgwY-BfqG-knNkEm

LVM & Linux


nico@dnarten: ~
nico@dnarten:~$ sudo lvdisplay
--- Logical volume ---
LV Path /dev/ubuntu-vg/ubuntu-lv
LV Name ubuntu-lv
VG Name ubuntu-vg
LV UUID c8X437-CKVq-CILG-wymL-zo00-35xx-yHePHg
LV Write Access read/write
LV Creation host, time ubuntu-server, 2019-04-13 19:10:09 +0200
LV Status available
open 1
LV Size <249.00 GiB
Current LE 63743
Segments 1
Allocation inherit
Read ahead sectors auto
- currently set to 256
Block device 253:0

LVM & Linux

nico@dn10: ~

```
nico@dn10:~$ sudo lvdisplay
  --- Logical volume ---
  LV Path                /dev/dn03-vg/root
  LV Name                root
  VG Name                dn03-vg
  LV UUID                EBHPX0-dLk4-d0zx-3tc3-yBq6-eaJ3-RY5tfE
  LV Write Access        read/write
  LV Creation host, time dn03, 2019-03-24 07:12:24 +0100
  LV Status              available
  # open                 1
  LV Size                <135.02 GiB
  Current LE             34565
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0

  --- Logical volume ---
  LV Path                /dev/dn03-vg/swap_1
  LV Name                swap_1
  VG Name                dn03-vg
  LV UUID                zVIkve-ojw1-UYk1-heK9-cpXO-efe2-f4scqc
  LV Write Access        read/write
  Creation host, time    dn03, 2019-03-24 07:12:24 +0100
  tatus                  available
  pen                    2
  Size                   976.00 MiB
  Current LE             244
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
```