Lesson #4
Creation and basic management of storage for OS and user data

Advanced Linux Administration

Robin Hack

Introduction

30 min

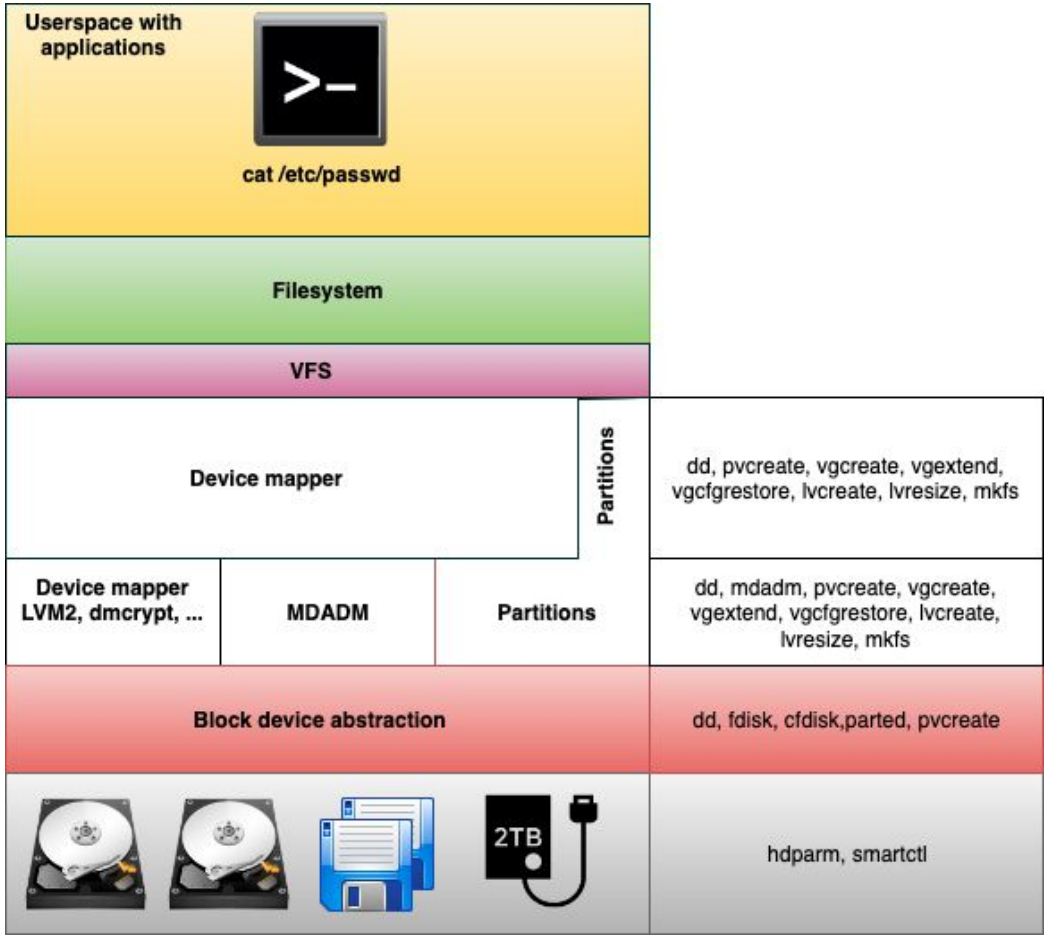
Trivial task…

- Get content of file

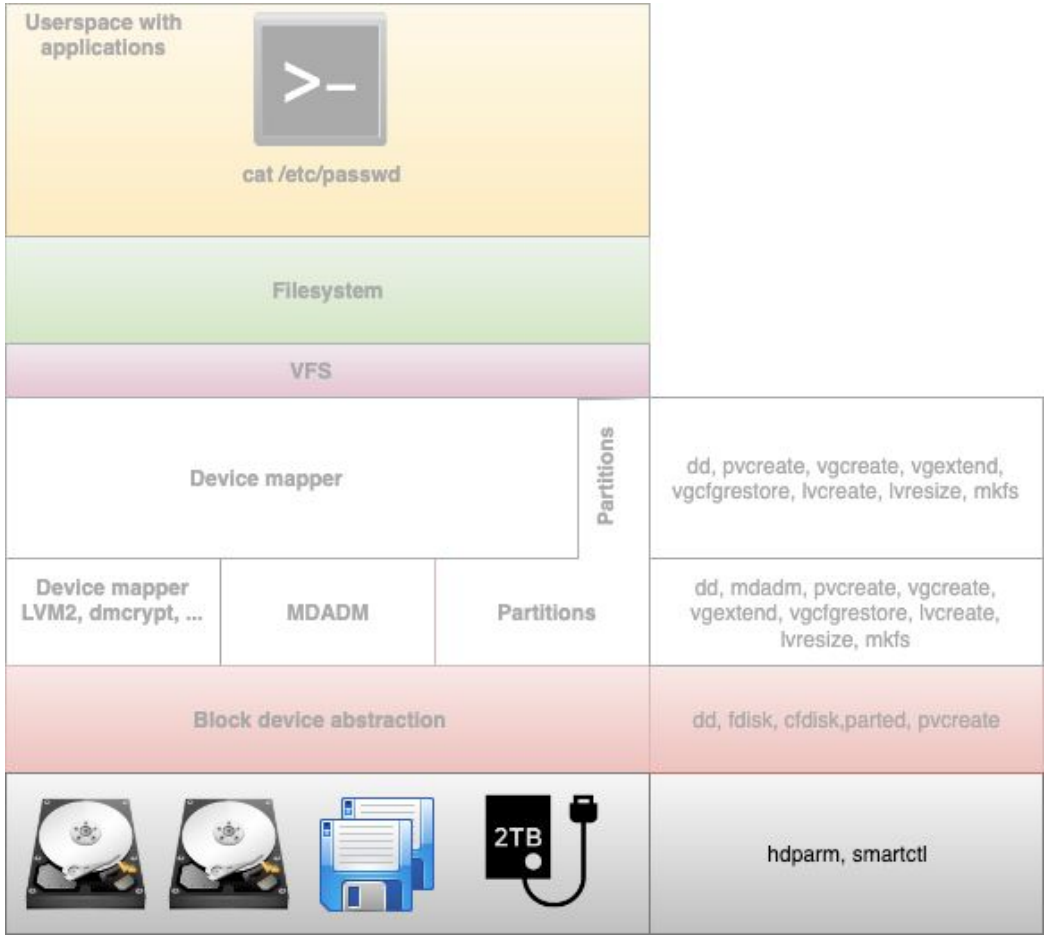
```
# cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
....
```

Trivial task…



HW

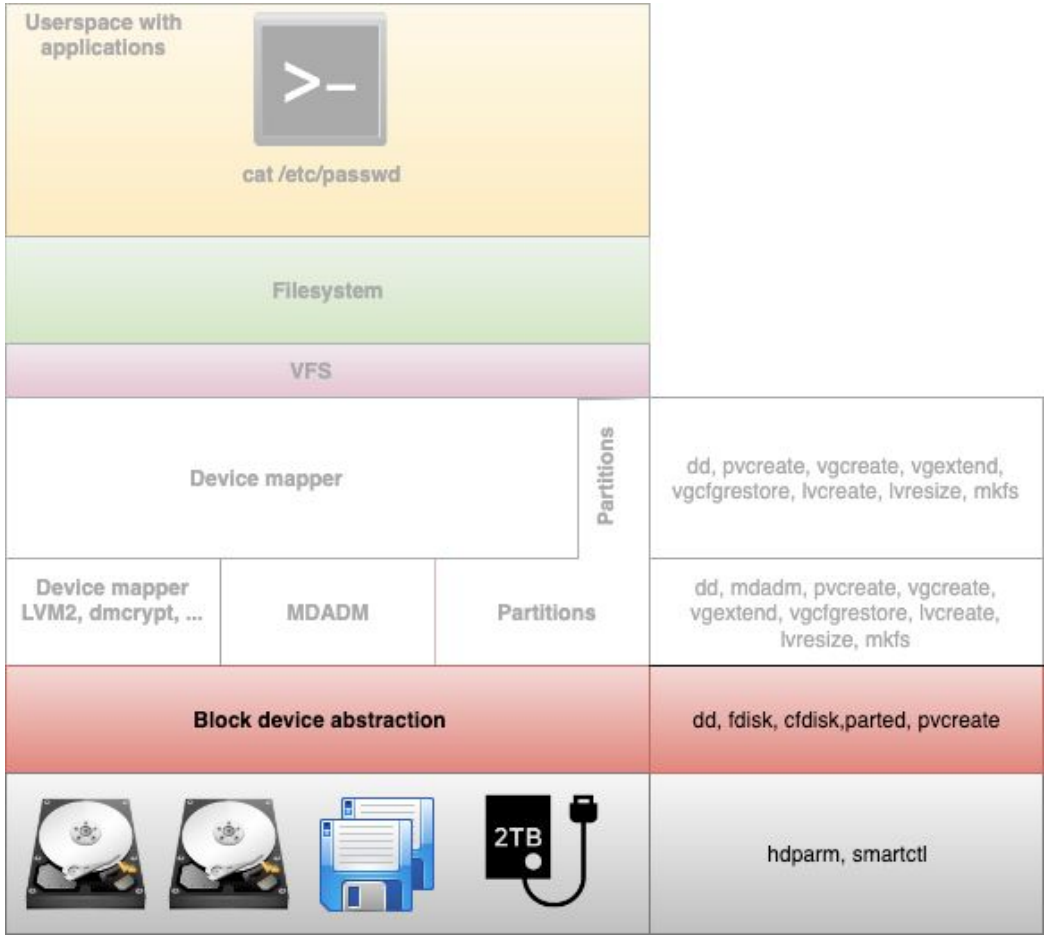

Userspace with applications
cat /etc/passwd
Filesystem
VFS
Device mapper
Partitions
dd, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Device mapper LVM2, dmcrypt, ...
MDADM
Partitions
dd, mdadm, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Block device abstraction
dd, fdisk, cfdisk,parted, pvcreate
2TB
hdparm, smartctl

Low level: hardware and communication

- Different protocols... SAS, SATA, ATA-IDE, SCSI, NVMe ...
- Different hardware layouts ... NAND/NOR flash, magnetic surface, ...

Solved by hardware and operating system abstraction.

Block Layer


Userspace with applications
cat /etc/passwd
Filesystem
VFS
Device mapper
Partitions
dd, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Device mapper LVM2, dmcrypt, ...
MDADM
Partitions
dd, mdadm, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Block device abstraction
dd, fdisk, cfdisk,parted, pvcreate
2TB
hdparm, smartctl

OS will save you with block/character layer abstraction

Block layer abstraction

- Physical:
  - /dev/sda, /dev/hda, /dev/fd0, /dev/sr0
- Virtual
  - /dev/md0, /dev/mapper/work
  - network block devices

Easy to use! (sort of and (almost) everything is file in unix)

```
# dd if=/dev/sda of=image.png skip=1024 ibs=1024 obs=4567 count=1
# file image.png
image.png PNG image data, 1440 x 900, 8-bit/color RGBA,
non-interlaced
```

Block device abstraction - special devices - loop device

"In Unix-like operating systems, a loop device, ... is a pseudo-device that makes a file accessible as a **block device**."

https://en.wikipedia.org/wiki/Loop_device

How to create loop device (easy and nice way):

```
# dd if=/dev/zero of=~/file.img bs=1MiB count=10
# losetup --find --show ~/file.img
/dev/loop0
```

Character device abstraction - special devices - zero & null

/dev/zero

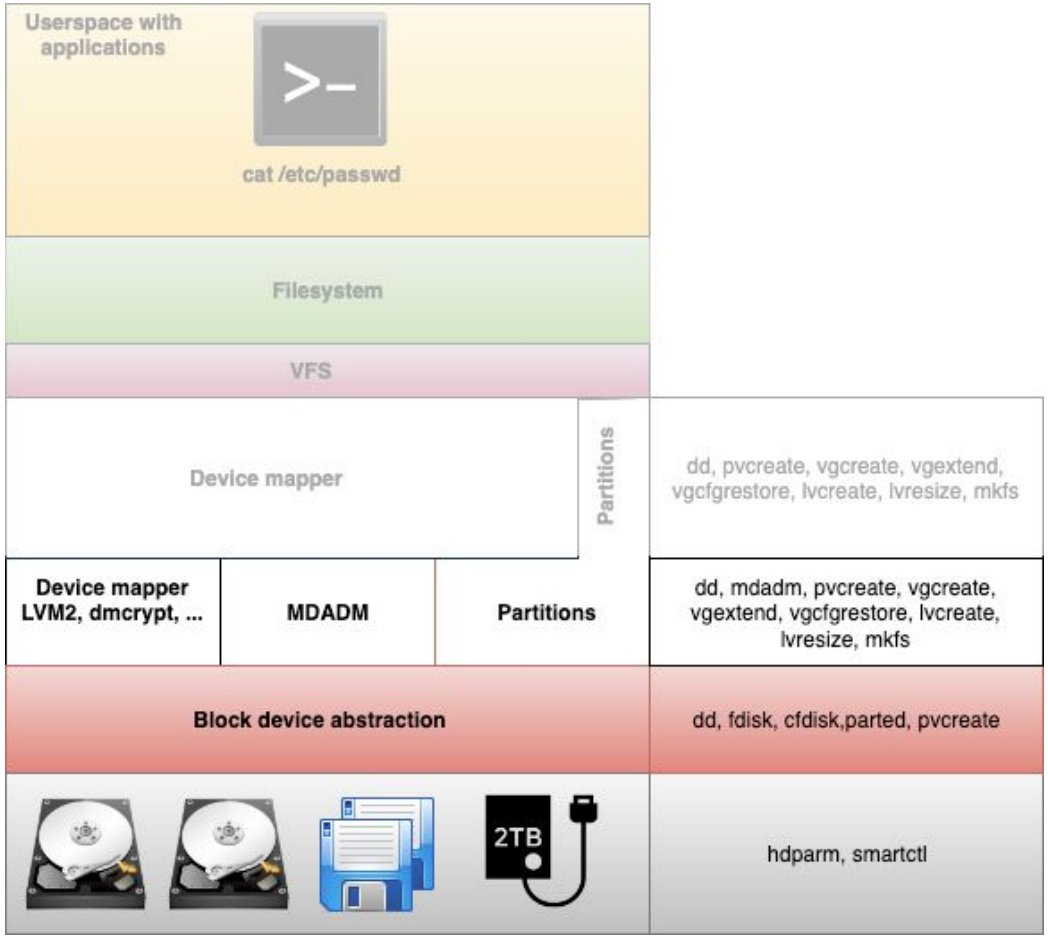
"/dev/zero is a special file (**character device**) in Unix-like operating systems that provides as many null characters (ASCII NUL, 0x00) as are read from it."

https://en.wikipedia.org/wiki//dev/zero

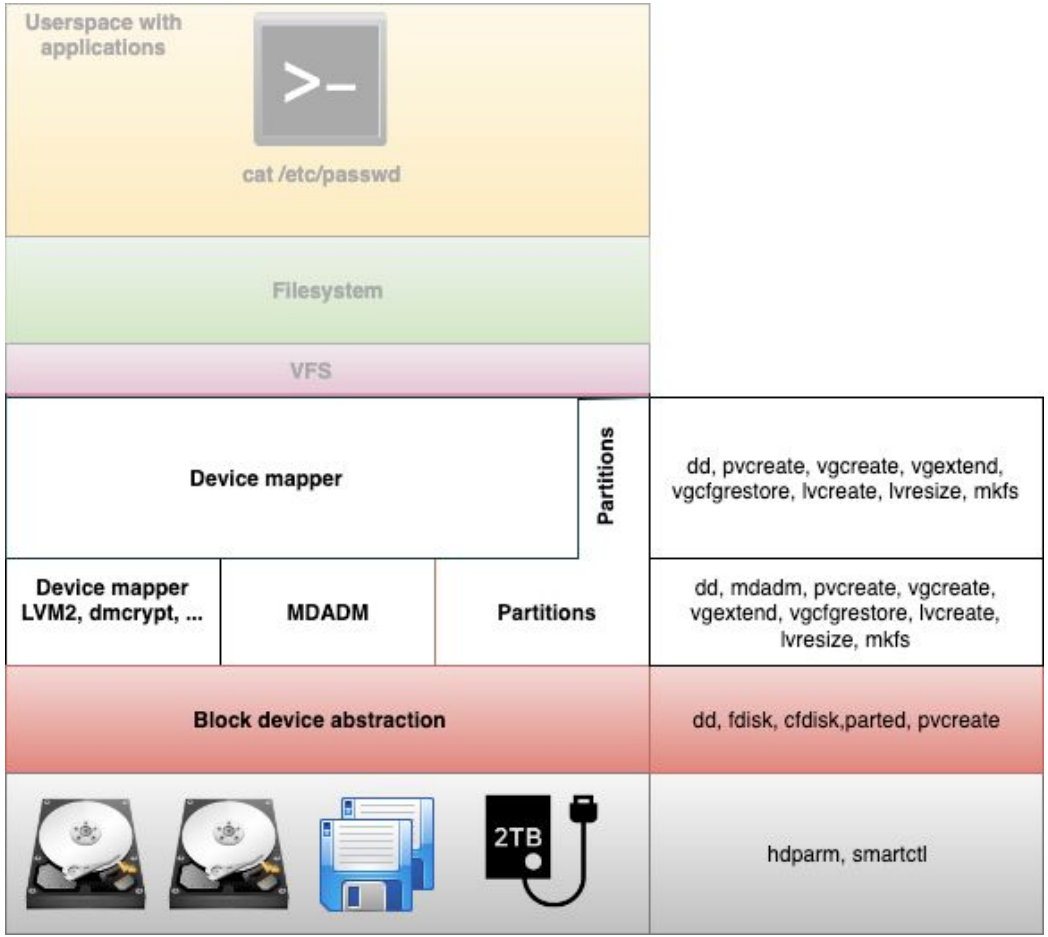
/dev/null

"In some operating systems, the null device is a device file that discards all data written to it but reports that the write operation succeeded."
https://en.wikipedia.org/wiki/Null_device

Partitions LVM



Partitions LVM



2 types of partitioning & tools

MBR partitioning

- `fdisk`
- `cfdisk`
- `parted`

Useful tools and commands:

- `lsblk -o +UUID`
- `cat /proc/partitions`
- `ls -la /dev/disk/by-uuid/`

LVM:

- `pvcreate`
- `pvremove`
- `vgcreate`
- `lvcreate`
- `lvremove`
- `lvextend`
- `vgremove`
- `vgdisplay`
- `lvdisplay`

MBR partitioning

Still in use and necessary for some use cases

- Multi-boot (OSes which doesn't support UEFI)
- Separate /boot partition (old systems, rest of the system encrypted)
- USB flash drives with multiboot
- Single board computers (rpi, bananapi, orangepi, ...)

Cons:

- Not flexible - when you cut your device into parts, you basically lost ability to change it again or it's difficult

fdisk

```
root@bananas:/# fdisk /dev/sda

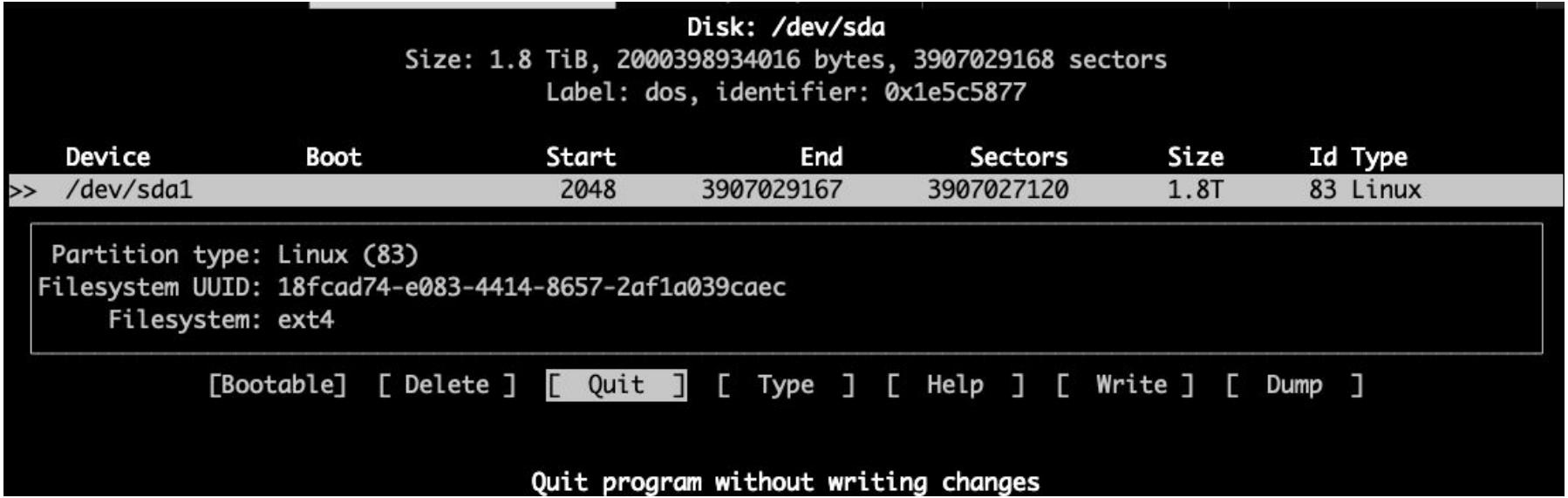
Welcome to fdisk (util-linux 2.29.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p
Disk /dev/sda: 1.8 TiB, 2000398934016 bytes, 3907029168 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: dos
Disk identifier: 0x1e5c5877

Device     Boot Start        End    Sectors  Size Id Type
/dev/sda1        2048 3907029167 3907027120  1.8T 83 Linux

Command (m for help):
```

cfdisk

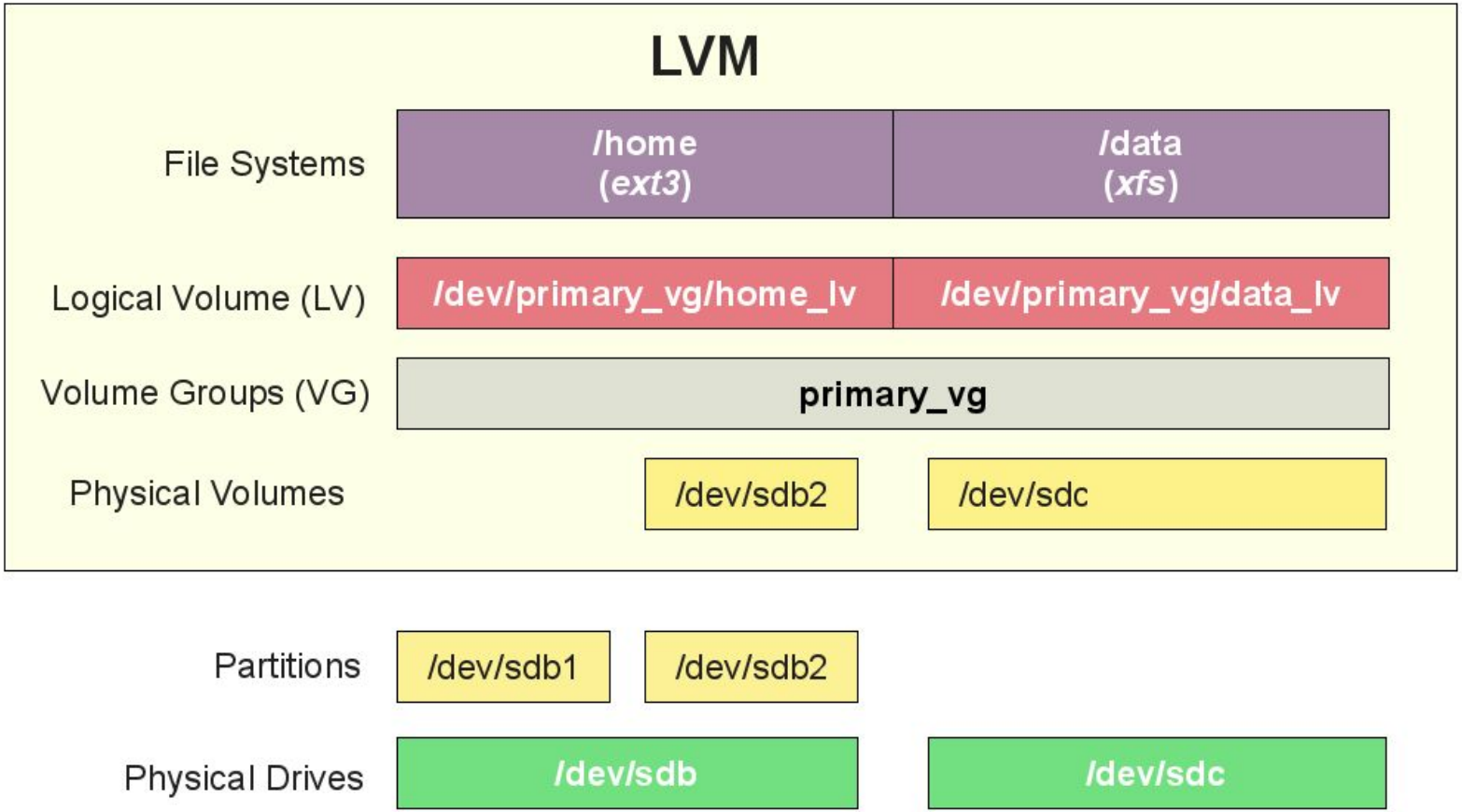

Disk: /dev/sda
Size: 1.8 TiB, 2000398934016 bytes, 3907029168 sectors
Label: dos, identifier: 0x1e5c5877
Device Boot Start End Sectors Size Id Type
>> /dev/sda1 2048 3907029167 3907027120 1.8T 83 Linux
Partition type: Linux (83)
Filesystem UUID: 18fcad74-e083-4414-8657-2af1a039caec
Filesystem: ext4
[Bootable] [Delete] [Quit] [Type] [Help] [Write] [Dump]
Quit program without writing changes

Logical Volume Manager (LVM)

LVM provides a method of allocating space on storage devices that is more flexible than MBR.

Pros:

- Flexible - you can add more space from more devices
- One (or more) big poll of space which can be sliced
- Stripped, mirrored and snapshot volumes
- RAID ability


LVM
File Systems
/home
(ext3)
/data
(xfs)
Logical Volume (LV)
/dev/primary_vg/home_lv
/dev/primary_vg/data_lv
Volume Groups (VG)
primary_vg
Physical Volumes
/dev/sdb2
/dev/sdc
Partitions
/dev/sdb1
/dev/sdb2
Physical Drives
/dev/sdb
/dev/sdc

LVM - basic tools and usage - `pvcreate`

`pvcreate` initializes Physical Volume for later use by the Logical Volume Manager (LVM). Each Physical Volume can be a disk partition, whole disk, meta device, or loop file.

Usage:

```
# pvcreate /dev/loop0
# pvcreate /dev/loop1
```

LVM - basic tools and usage - `vgcreate`

`vgcreate` creates a new Volume Group using the block special device or devices.

Usage:

```
# vgcreate vgtest /dev/loop0 /dev/loop1

# vgdisplay
```

LVM - basic tools and usage - `lvcreate`

`lvcreate` creates a new Logical Volume in a Volume Group

Usage:

```
# lvcreate vgtest -n work -L400M

# lvdisplay
```

LVM - basic tools and usage - lvextend

`lvextend` allows you to extend the size of a Logical Volume.

All operations must run in an "offline mode". It means that filesystem must be unmounted.

```
# lvextend -L +400M /dev/vgtest/work
# lvdisplay
```

```
# resize2fs /dev/vgtest/work
```

The resize2fs program will resize ext2, ext3, or ext4 file systems.

Software RAID

What is RAID?

Software RAID

RAID (Redundant Array of Inexpensive Disks or Drives, or Redundant Array of Independent 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.

Pros:

- No need for special hardware
- Cheap
- Reliable

Software RAID - how to create - mdadm

Prepare loop devices:

```
# dd if=/dev/zero of=disk0 bs=500M count=1
# cp disk0 disk1
# for p in {0..1}; do losetup "loop$p" "disk$p"; done
```

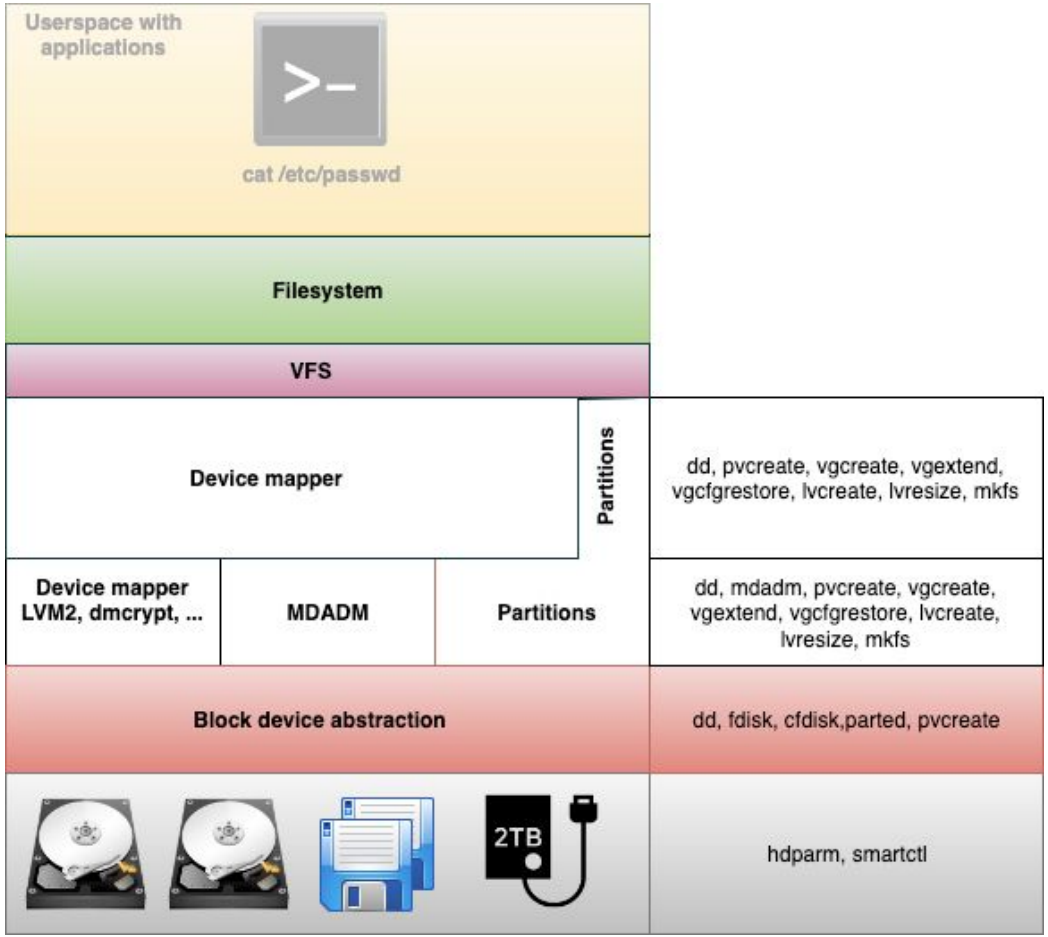
Create level 1 md0 device:

```
# mdadm --create /dev/md0 --level=mirror --raid-devices=2 /dev/loop0 /dev/loop1
```

Show changes:

```
# cat /proc/mdstat
# mdadm --detail /dev/md0
# mkfs.ext4 /dev/md0
```

File system


Userspace with applications
cat /etc/passwd
Filesystem
VFS
Device mapper
Partitions
dd, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Device mapper LVM2, dmcrypt, ...
MDADM
Partitions
dd, mdadm, pvcreate, vgcreate, vgextend, vgcfgrestore, lvcreate, lvresize, mkfs
Block device abstraction
dd, fdisk, cfdisk,parted, pvcreate
2TB
hdparm, smartctl

File systems - what is a file system?

What do you think?

File systems - what is a file system

"In computing, a file system or filesystem (often abbreviated to fs), controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins."

Wikipedia - File system

File systems - differences

- Maximum file system size (ext4: 1 EiB)
- Maximum file size (ext4: 16 TiB)
- File name length (ext4: 255 chars)
- Allowed chars in file name (ext4: All bytes except NUL ('\0') and '/')
- Maximum number of files (can be configured sometimes)
- Access control lists
- Failure resistance (journaling, soft updates)
- Tooling
- Licence or patents
- Underlying hardware (Flash file system)

For more info: Wikipedia - Comparison of file systems

File systems - tons and tons

APFS, HAMMER, reiserfs, ext{1..4}, fat{16..32}, squashfs, UFS, JFS, XFS, ...

Not all file systems are supported by Linux kernel.
License reasons: incompatible licence with linux kernel licence.
Partial support: read only.

Most used and know from my POV in Linux OS:

- ext4 (default on most distributions for root / & /boot partition)
- btrfs (default from Fedora Workstation 33)
- XFS (default for Fedora Server since version 22)
- *FAT - FAT compatible (vfat, extfat, fat32, ...)
- zfs (used for big data)

File systems - how to create

```
# mkfs.ext4 /dev/loop0

# mkfs -t ext4 /dev/loop0
```

File systems - how to use

```
# mkdir /mnt/fs

# mount /dev/loop0 /mnt/fs
```

File systems - where is my space?!

```
# dd if=/dev/zero of=disk.img bs=500M
524288000 bytes (524 MB, 500 MiB) copied, 23.6756 s, 22.1 MB/s

# ... create loop device ...
# mkfs.ext4 /dev/loop0
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

File systems - where is my space?!

```
# mount /dev/loop0 /mnt/fs
# df -h
/dev/loop0     477M  2.3M  445M   1% /mnt/fs
```

File systems can have:

- Need to hold its own metadata (journal, superblocks, informations about inodes, directory names, ...)
- Different block size => smallest possible allocatable unit
- Different units used.. MB vs MiB
- Space reserved for root user (default is 5% on ext4)

File systems - /etc/fstab

Mounting are not persistent operation!

```
# mount /dev/loop0 /mnt/fs
```

will not survive reboot!

File systems - /etc/fstab

"The fstab (or file systems table) file is a system configuration file commonly found at **`/etc/fstab`** ...".
https://en.wikipedia.org/wiki/Fstab

```
$ man 5 fstab
```

Format:

```
# device identificator  destination type of fs  mount flags         dump fsck
/dev/sda2               /home       ext4        defaults,noexec     0     1
LABEL=home2             /home2       ext4        defaults,noexec    0     1
UUID=a8f32dfe-cc97-4ba7-9089-c6501f70a92e /home3 ext4 defaults      0     1
```

How to determine UUID or LABEL? Tools: **`blkid`** or **`lsblk`**

File systems - /etc/fstab

Is it possible to mount loop devices via fstab? Yes!

```
# dd if=/dev/zero of=/root/disk.img bs=400M count=1
# echo "/root/disk.img /mnt/fs ext4 defaults 0 0" >> /etc/fstab
```

Workshop

60 min

Workshop labs

- In the following next slides there are 6 labs total (5 normal + 1 bonus)
- Each lab has a time estimate how much time should you spend on it if everything goes well
- We encourage you to help each other or rise your hand to get help from lecturers
- If you couldn’t finish all labs during this class, you should complete them on your own later because learned skills will be used in following lectures or during a final practical exam.
- *NOTE: Focus on the lab content and leave any exploration or deep dive desires as a self-study for later.*

Workshop labs - before you begin

- Best way how to clean up your environment after labs is to use snapshot of whole virtual machine (fast way).
- Manually you can use:
  - Detach loop device or all devices:
    - losetup -d /dev/loop0
    - losetup -D
  - Stop software raid:
    - mdadm --stop /dev/md0
  - Remove logical volume
    - lvremove /dev/vgtest/lvtest
  - Remove volume group
    - vgremove vgtest
  - Remove pv group
    - pvremove /dev/loop0

Workshop labs - troubleshooting

- Some operations must be done in order.
- For example:
  - you have mounted filesystem on loop device to `/mnt/fs`
  - you want to remove loop device 0
  - `losetup -d /dev/loop0` - works!
  - `losetup /dev/loop0 disk2.img` - doesn't work :(.
  - Reason: losetup is lazy -> device is really gone only when all references are removed
  - umount `/mnt/fs`
  - `ls -la /dev/loop0` - device is still present! But is free to use.
  - `losetup /dev/loop0 disk2.img` - works \o/
  - you can always check state with `losetup` (without any params)

Workshop labs - cleanup example - loop device

```
# losetup --find --show disk.img
/dev/loop4
# mkfs.ext4 /dev/loop4
# mount /dev/loop4 /mnt/

# losetup
NAME       SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE
/dev/loop4 0         0      0         0 /home/rhack/disk.img

# umount /dev/mnt
# losetup -d /dev/loop4
```

Labs 1/2

- **Lab 1** - BASIC FILESYSTEM (10 min)
  - create loop device, create filesystem, mount filesystem
- **Lab 2** - SOFTWARE RAID (10 min)
  - create loop devices, create software raid, create filesystem, mount filesystem
- **Lab 3** - LVM (10 min)
  - create loop devices, create volume group, create logical volume, create filesystem, mount
- **Lab 4** - LVM - crash course (15 min)
  - create loop devices, create volume group, create logical volume, create filesystem, resize logical volume to unsane value, fix previous issue
- **Lab 5** - SOFTWARE RAID - crash course (15 min)
  - create loop devices, create software raid of level 1, create filesystem, replace faulty storage device

Work on the lab from terminal as a root user: `$ sudo -i` or `$ su -`

Labs 2/2

- **Lab 6** - BONUS - persistency

Work on the lab from terminal as a root user: `$ sudo -i` or `$ su -`

Lab 1 - BASIC FILESYSTEM (10min)

Lab1 - Intro

- Create loop device
- Create ext4 filesystem
- Mount filesystem to `/mnt/mytest`

Don't forget to cleanup your environment

Lab 2 - SOFTWARE RAID (10min)

Lab2 - Intro

- Create 2 loop devices.
- Create raid 1 device composed with 2 loop devices
- Create filesystem on raid device
- Mount filesystem to `/mnt/mytest2`

Don't forget to cleanup your environment

Lab 3 - LVM
(10min)

Lab3 - Intro

- Create 2 loop devices
- Create volume group named `vgtest` on top of loop devices
- Create logical volume named `work`
- Create filesystem on logical volume with name `work`
- Mount filesystem to `/mnt/mytest3`
- Check with: `lsblk, lvdisplay, vgdisplay`

Lab3 - 1/5 - Create loop devices loop0 and loop1

Create loop0 a loop1 block device:

```
# dd if=/dev/zero of=disk0 bs=500M count=1

# cp disk0 disk1

# losetup loop0 ./disk0

# losetup loop1 ./disk1
```

Or you can use:

```
# for p in {0..1}; do losetup "loop$p" "disk$p"; done
```

Lab3 - 2/5 - Mark devices for LVM use

```
# pvcreate /dev/loop{0..1}
```

Lab3 - 3/5 - Create volume group

```
# vgcreate vgtest /dev/loop{0..1}

# vgdisplay
```

Lab3 - 4/5 - Create logical volume

```
# lvcreate vgtest -n work -L400M

# lvdisplay
```

Lab3 - 5/5 - Create filesystem, mount filesystem

```
# mkdir /mnt/mytest3

# mkfs.ext4 /dev/vgtest/work

# mount /dev/vgtest/work /mnt/mytest3
```

Lab 4 - LVM - crash course (15min)

Lab4 - Intro

- Create 2 loop devices
- Create volume group named `vgtest` on top of loop devices
- Create logical volume named `work`
- Create file system on logical volume
- Resize logical volume to bigger size
- Resize logical volume to smaller "insane" size
- Undo possible issue

Lab4 - 1/8 - Create loop devices loop0 and loop1

Create loop0 a loop1 block device:

```
# dd if=/dev/zero of=disk0 bs=500M count=1

# cp disk0 disk1

# losetup loop0 ./disk0

# losetup loop1 ./disk1
```

Or you can use:

```
# for p in {0..1}; do losetup "loop$p" "disk$p"; done
```

Lab4 - 2/8 - Mark devices for LVM use

```
# pvcreate /dev/loop{0..1}
```

Lab4 - 3/8 - Create volume group; create logical volume

```
# vgcreate vgtest /dev/loop{0..1}

# vgdisplay vgtest

# lvcreate vgtest -n work -L400M

# lvdisplay vgtest
```

Lab4 - 4/8 - Create filesystem, mount filesystem, umount filesystem

```
# mkdir /mnt/mytest4

# mkfs.ext4 /dev/vgtest/work

# mount /dev/vgtest/work /mnt/mytest4

# mount | grep mytest4

# lsblk

# umount /mnt/mytest4
```

Lab4 - 5/8 - Resize volume to bigger size

Resize your logical volume to bigger size.

```
# lvextend -L +200M  /dev/vgtest/work
```

What about filesystem?

```
# resize2fs /dev/vgtest/work
```

Lab4 - 6/8 - Do some not nice changes to logical volume

Resize your logical volume to smaller size. Ignore warnings.

```
# lvresize /dev/vgtest/work -L200M
```

Lab4 - 7/8 - Everything is lost… or not?

Take a look at metadata backup.

```
# vgcfgrestore --list vgtest

File: /etc/lvm/archive/vgtest_00014-478352324.vg

VG name:        vgtest

Description: Created *before* executing 'lvresize /dev/vgtest/work
-L200M'

Backup Time:    Wed Sep 18 16:52:04 2019
```

Lab4 - 8/8 - Nothing is lost! Undo your changes.

Apply changes.

```
# vgcfgrestore vgtest \
  -f /etc/lvm/archive/vgtest_00014-478352324.vg
```

Don't forget to cleanup your environment

Lab 5 - SOFTWARE RAID crash course (15min)

Lab5 - Intro

- create 2 loop devices
- create software raid 1 on top of loop devices
- mark device as failed and remove it from raid array
- prepare new loop device as a replacement
- add new loop device to array
- check for changes

Lab5 - 1/5 - Create loop devices loop0 and loop1

Create loop0 a loop1 block device:

```
# dd if=/dev/zero of=disk0 bs=500M count=1

# cp disk0 disk1

# losetup loop0 ./disk0

# losetup loop1 ./disk1
```

Or you can use:

```
# for p in {0..1}; do losetup "loop$p" "disk$p"; done
```

Lab5 - 2/5 - Create software raid of level 1

Create level 1 md0 device:

```
# mdadm --create /dev/md0 --level=mirror \
    --raid-devices=2 /dev/loop0 /dev/loop1
```

Show changes:

```
# mdadm --detail /dev/md0
```

Lab5 - 3/5 - Degrade array

Now, you just need to degrade raid array.

```
# mdadm --manage /dev/md0 --fail /dev/loop0

# mdadm --detail /dev/md0

# mdadm --manage /dev/md0 --remove /dev/loop0

# mdadm --detail /dev/md0
```

Lab5 - 4/5 - Prepare new storage

First, you need new disk as a replacement of faulty one

```
# dd if=/dev/zero of=disk-replace bs=500M count=1

# losetup /dev/loop4 disk-replace
```

Lab5 - 5/5 - Add new disk and watch for changes

```
# mdadm --manage /dev/md0 --add /dev/loop4

# mdadm --detail /dev/md0
```

Don't forget to cleanup your environment

Lab 6 - BONUS - persistency (5min?)

Lab6 - Bonus

- Create loop device
- Create filesystem on loop device
- Create directory `/mnt/persistent-loop`
- Make sure that `/mnt/persistent-loop` mount point will survive reboot
  - check with `umount /mnt/persistent-loop`
  - `# mount -a`
  - You should have /mnt/persistent-loop mounted without any issues

Mid-semester assignment (20 points)

Instructions for submission

- Your final solution has to be uploaded to the FIT system in a file named `xloginXX-fit-ili.sh`
- Script must run on a clean, default installation of Fedora Server. Install any required RPMs not present after default installation at the beginning of your script so it can run properly.
- Script must be written in Bash and have right end of lines format.
- During evaluation of your solution script will be executed under `root` user without any additional command line options. (You can implement your own command line options, but they won't be used during evaluation of the 10 requirements.)
- Script must log (e.g. using command echo) a simple description of actions for the 10 requirements to a standard output, for example:
  - ```
    1) Creating 4 loop devices
    .. creating file number 3
    ```
- Script does not have to clean-up after itself, but it is recommended to use a snapshotting in your virtual machine to have a clean environment for consecutive executions of your script.
    ```

Mid-semester Assignment (20 points)

Prepare an executable Bash script performing the following requirements:

1) Vytvořte 200 MB soubor /var/tmp/ukol.img. (1b)
2) Vytvořte loop device pro ukol.img. (1b)
3) Vytvořte filesystem na nově vytvořeném loop device (ext4). (1b)
4) Upravte /etc/fstab tak, aby se filesystem připojoval (mount) automaticky (pomocí loop device). (2b)
5) Nově vytvořený filesystem připojte (namountujte) do /var/www/html/ukol. (1b)
6) Pomocí nástroje yum stáhněte ze systémových repozitářů balíčky předané jako argumenty skriptu a uložte do /var/www/html/ukol. (2b)
7) Vygenerujte repodata v /var/www/html/ukol. (1b)
    - **DŮLEŽITÉ:** Následně upravte selinux context v /var/www/html/ukol příkazem: restorecon -Rv /var/www/html/ukol.
8) Nakonfigurujte /etc/yum.repos.d/ukol.repo tak, aby přistupoval k repozitáři “ukol” přes URL: http://localhost/ukol.(2b)
9) Nainstalujte a spusťte webový server pomocí služby httpd (Apache). (2b)
10) Vypište dostupné yum repozitáře, abyste ověřili, že repozitář "ukol" je dostupný. (2b)
11) Proveďte odpojení (unmount) filesystemu připojeného do /var/www/html/ukol (1b)
12) Následně ve skriptu spusťte příkaz “mount -a” a ověřte, že filesystem na loop device je připojen do /var/www/html/ukol. (2b)
13) Pomocí příkazu yum vypište informace (info) o dostupných balíčcích, přičemž při volání příkazu yum zakažte všechny repozitáře kromě repozitáře "ukol". (2b)

Links and Resources

LVM 101

How to Extend/Reduce LVM's (Logical Volume Management) in Linux – Part II

RAID on wikipedia

Comparison of file systems

How to Set Up Software RAID 1 on an Existing Linux Distribution