

# Linux, day 9



# Objectives covered

Objective	Summary	Book
1.1	Storage concepts	11
1.3	Given a scenario, manage storage in a Linux environment	11
2.1	PKI certificates	16

# Homework = crucial

- **Prepare** by reading the indicated pages.
- **Repeat** by doing the extra labs.
- **Test** by doing the knowledge-checks.

# LAB: Partitioning



# New disks for our VM!

- Using VirtualBox, let's add three disks!
  - Create three new, "thin provisioned" disks.
  - Each should be 100MB.
  - We will use these in next labs & classes too!
- After we reboot, let's see if they're found!

# Are they there?

- Do we see new devices in /dev?
- Does *dmesg* (or *journalctl -b*) show new disks?

```
$ journalctl -b | grep -i disk
```

```
$ ls /dev/sd* /dev/vd*
```

# Partitioning: gparted

- We will need the graphical environment.
- Start the *gparted* tool.
  - The pull-down shows available disks.
  - It even helps you format the partition!
  - Changes only happen with the "APPLY" button.

# Partitioning: fdisk

- "*fdisk*" does partitioning from the CLI.
  - You can use it through SSH!
- Its usage is tricky and uses one-letter commands.
- Let's demo!



# fdisk commands

<u>Command</u>	<u>Function</u>
m	Help / list of commands
g	Create new GPT partition table.
p	Print / show the partition table.
n, d	Add, or remove, a partition.
l, t	Show partition types, set partition type.
w	Write partition table to disk (" <i>APPLY</i> ").

# Are they there?

- Do we see new partitions in /dev?
- Does *dmesg* (or *journalctl*) show new devices?

```
$ journalctl --since "10 minutes ago"
```

```
$ ls -lrt /dev/sd*
```

# Recap: drives and partitions

- All devices are in */dev/*.
  - *hd\_, sd\_, nvme\_, vd\_*.
- Drives get a letter, partitions a number.
  - *hda1, sdb2, nvme1, vdd1*

# LAB: Formatting



# Commands per FS type

- See what you find!
- Type "*mkfs*" and press <TAB> twice.

<code>mkfs</code>	<code>mkfs.cramfs</code>	<code>mkfs.ext3</code>
<code>mkfs.fat</code>	<code>mkfs.msdos</code>	<code>mkfs.xfs</code>
<code>mkfs.btrfs</code>	<code>mkfs.ext2</code>	<code>mkfs.ext4</code>
<code>mkfs.minix</code>	<code>mkfs.vfat</code>	

# Mount points and formatting

- Let's make mount points and format our disks.

```
$ sudo mkdir /mnt/b /mnt/c /mnt/d
```

```
$ sudo mkfs.ext4 /dev/sdb1
```

```
$ sudo mkfs.xfs /dev/sdc1
```

```
$ sudo mkfs.vfat /dev/sdd1
```

# Mounting!

- We can load the file systems, but it's not permanent.

```
$ sudo mount -t ext4 /dev/sdb1 /mnt/b  
$ sudo mount -t xfs /dev/sdc1 /mnt/c  
$ sudo mount -t vfat /dev/sdd1 /mnt/d
```

# Mounting at boot

- We can add these to `"/etc/fstab"`.
  - Make a backup copy of this file first!

```
/dev/sdb1  /mnt/b  ext4  defaults 1 3
/dev/sdc1  /mnt/c  xfs   defaults 1 3
/dev/sdd1  /mnt/d  vfat  defaults 1 3
```



# Re-mounting

- Let's drop the mounts and re-mount them as test.
- Reboot. Do the mounts re-appear?

```
$ sudo umount /mnt/b /mnt/c /mnt/d  
$ sudo mount --all      # fix any errors!  
$ sudo reboot
```

# Let's break stuff!

- After making the new FS, adjust the perms!
  - Make them writable for your group/account.
  - This will fail for *vfat*. Why?

```
$ sudo chgrp tess /mnt/b /mnt/c /mnt/d  
$ sudo chmod 775 /mnt/b /mnt/c /mnt/d
```

# Let's break stuff

```
$ for i in {1..1000}  
do touch /mnt/b/$i; done
```

```
$ ls /mnt/b/
```

# Now we'll break it for real

```
$ sudo dd if=/dev/urandom of=/dev/sdb1 \
bs=1M count=100
```

```
$ ls /mnt/b/
```

```
$ journalctl --since '2 minutes ago'
```

# LAB: FUSE and sshfs



# FUSE: filesystem in userspace

- Sometimes you need a "weird" file system,
  - But don't want a kernel module for it.
  - You just want a quick integration into Linux.
- FUSE makes this possible

See also: [FUSE on ArchWiki](#)

# FUSE: Some examples

- Gitfs - Git repository with auto-commit
- Sshfs - Remote directory over SSH
- Gdrivefs - Google Drive
- NTFS3G - Windows hard drives
- Adbfs - Android devices, over USB

# Let's FUSE our VMs...

- On the Ubuntu VM:

```
$ sudo apt install sshfs
```

```
$ sudo mkdir /mnt/fedoratmp
```



# Let's FUSE our VMs...

- On the Ubuntu VM:

```
$ sudo sshfs -o \
allow_other,default_permissions \
tess@fedora:/tmp /mnt/fedoratmp
```

```
$ ls /mnt/fedoratmp
```

# Outcome

- You just mounted /tmp from Fedora,
  - Onto /mnt/fedoratmp on Ubuntu,
  - Via SSH!
- You can make files, change them, etc.
  - Without using "normal" file share protocols.

# Advanced practice

- Can you make this work with `/etc/fstab`?
  - Prove it!

# Closing



# Homework

- Reading:
  - Chapter 11.

# Homework

- Fix your `/dev/sda`, `sdb` and `sdc`.
- Reformat them.
- Change the entries in `/etc/fstab`, to use:
  - `/mnt/a: by-uuid`
  - `/mnt/b: by-partuuid`
  - `/mnt/c: by-label`.

# Homework

- Go do:
  - Make a 5GB VirtualBox disk image,
  - Formatted as XFS,
  - Which gets mounted on */var/appdata/* at boot.

# Reference materials





# Resources

- [PowerCert NAS vs SAN](#) (YouTube)
- [Standard RAID levels](#) (Wikipedia)
- [Synology RAID calculator](#)
- [Setting up raw devices for Oracle](#)
- [An introduction to storage terminology](#)
- [RHEL 8: XFS Copy-on-write](#)

# Resources

- [Snapshots 101: CoW vs RoW](#)
- [Changing a file system's UUID](#)
- [Persistent block device naming](#)
- [A Linux user's guide to LVM](#)
- [Using SSHFS to mount a remote directory](#)
- [A few FUSE filesystems](#) (ArchWiki)