8.1 Speicher, Mounts und Dateisysteme

Linux verwaltet Datenträger, Partitionen, Dateisysteme und eingebundene Speicherbereiche anders als Windows.

Es gibt keine Laufwerksbuchstaben wie C: oder D:. Stattdessen werden Dateisysteme in den gemeinsamen Verzeichnisbaum eingehängt.

Dieses Einhängen nennt man:

```
Mounten
```

Für Fachinformatiker Systemintegration ist dieses Thema wichtig, weil Server, NAS-Systeme, Backups, Container, Logdateien, Datenbanken und Freigaben sehr häufig von korrektem Speicherplatz, Mountpoints und Dateisystemen abhängen.

Merksatz:

```
Linux hängt Speicher in den Verzeichnisbaum ein.
```

Lernziele

Nach dieser Seite solltest du erklären können:

```
- was ein Datenträger ist
- was eine Partition ist
- was ein Dateisystem ist
- was ein Mountpoint ist
- wie man Speicherplatz prüft
- wie man Datenträger und Partitionen anzeigt
- was /etc/fstab macht
- was df,
  du,
  lsblk,
  mount
  und umount machen
- warum volle Dateisysteme Dienste stören können
- welche typischen Speicherfehler unter Linux auftreten
```

Grundidee

Ein Linux-System kann verschiedene Speicherbereiche nutzen:

```
- interne SSDs
- interne HDDs
- externe USB-Datenträger
- Netzlaufwerke
- NAS-Freigaben
- virtuelle Datenträger
- Container-Volumes
- RAM-Dateisysteme
- Backup-Speicher
```

Damit Daten dort gespeichert werden können, braucht man meistens:

```
1. Datenträger

2. Partition oder Volume

3. Dateisystem

4. Mountpoint

5. Einbindung ins System
```

Merksatz:

```
Datenträger allein reicht nicht.
Er muss passend partitioniert,
formatiert
und eingebunden sein.
```

Datenträger

Ein Datenträger ist das physische oder virtuelle Speichermedium.

Beispiele:

```
- SSD
- HDD
- USB-Stick
- SD-Karte
- virtuelles Laufwerk einer VM
- iSCSI-LUN
- Cloud-Block-Storage
```

Unter Linux erscheinen Datenträger häufig unter:

```
/dev
```

Beispiele:

```
/dev/sda

/dev/sdb

/dev/nvme0n1
```

Merksatz:

```
Datenträger erscheinen unter Linux oft als Gerätedateien unter /dev.
```

Partition

Eine Partition ist ein Bereich auf einem Datenträger.

Beispiel:

```
/dev/sda
```

ist der Datenträger.

```
/dev/sda1
```

ist die erste Partition auf diesem Datenträger.

Bei NVMe-Datenträgern sieht es oft so aus:

```
/dev/nvme0n1
```

Datenträger

```
/dev/nvme0n1p1
```

erste Partition

```
/dev/nvme0n1p2
```

zweite Partition

Merksatz:

```
Datenträger ist das Gerät.
Partition ist ein Bereich darauf.
```

Dateisystem

Ein Dateisystem legt fest, wie Dateien und Verzeichnisse auf einem Speicher organisiert werden.

Beispiele für Dateisysteme:

Dateisystem	Typischer Einsatz
ext4	sehr verbreitet unter Linux
XFS	häufig auf Servern
Btrfs	Snapshots, moderne Funktionen
ZFS	Integrität, Snapshots, Storage-Systeme
FAT32	einfache Wechselmedien
exFAT	größere Wechselmedien
NTFS	Windows-Dateisystem
tmpfs	Dateisystem im RAM

Merksatz:

```
Dateisystem = Organisationsform für Dateien auf Speicher.
```

Formatieren

Beim Formatieren wird auf einer Partition oder einem Volume ein Dateisystem erstellt.

Beispiel:

```
ext4-Dateisystem auf einer Partition erstellen
```

Wichtig:

```
Formatieren löscht normalerweise vorhandene Daten
oder macht sie nicht mehr direkt nutzbar.
```

Merksatz:

```
Formatieren ist gefährlich,
wenn Daten erhalten bleiben sollen.
```

Mountpoint

Ein Mountpoint ist ein Verzeichnis, an dem ein Dateisystem eingebunden wird.

Beispiele:

```
/mnt/backup

/media/usb

/srv/data

/var/lib/docker
```

Wenn ein Dateisystem dort gemountet ist, erscheinen seine Inhalte an dieser Stelle im Verzeichnisbaum.

Merksatz:

```
Mountpoint = Einhängepunkt im Verzeichnisbaum.
```

---

mount

Mit mount kann man anzeigen, welche Dateisysteme aktuell eingebunden sind.

Befehl:

```
mount
```

Gefiltert anzeigen:

```
mount | grep /mnt
```

Ein Dateisystem manuell einhängen:

```
mount /dev/sdb1 /mnt/backup
```

Wichtig:

```
Der Mountpoint muss existieren.
```

Beispiel:

```
mkdir -p /mnt/backup

mount /dev/sdb1 /mnt/backup
```

Merksatz:

```
mount hängt Dateisysteme ein oder zeigt Mounts an.
```

---

umount

Mit umount wird ein Dateisystem ausgehängt.

Beispiel:

```
umount /mnt/backup
```

oder:

```
umount /dev/sdb1
```

Wichtig:

```
Der Befehl heißt umount,
nicht unmount.
```

Wenn ein Dateisystem noch benutzt wird, kann umount fehlschlagen.

Mögliche Ursache:

```
Datei ist noch offen

Prozess nutzt das Verzeichnis

Shell befindet sich im Mountpoint
```

Merksatz:

```
umount hängt ein Dateisystem aus.
```

Warum umount fehlschlagen kann

Fehlerbild:

```
target is busy
```

Bedeutung:

```
Das Dateisystem wird noch verwendet.
```

Mögliche Ursachen:

```
Terminal befindet sich im Verzeichnis

Prozess liest oder schreibt Dateien

Dienst nutzt den Mount

Datei ist geöffnet
```

Prüfen kann man je nach System mit Werkzeugen wie:

```
lsof

fuser
```

Merksatz:

```
Ein benutzter Mount kann nicht sauber ausgehängt werden.
```

---

lsblk

lsblk zeigt Blockgeräte übersichtlich an.

Befehl:

```
lsblk
```

Mit Dateisysteminformationen:

```
lsblk -f
```

Typische Ausgabe enthält:

```
Name

Größe

Typ
```

```
Mountpoint

Dateisystem

UUID
```

Beispiel:

```
sda
├─sda1  ext4  /
└─sda2  swap
```

Merksatz:

```
lsblk zeigt Datenträger,
Partitionen
und Mountpoints.
```

---

df

df zeigt belegten und freien Speicherplatz von eingehängten Dateisystemen.

Häufiger Befehl:

```
df -h
```

Bedeutung:

Spalte	Bedeutung
Filesystem	Dateisystem oder Gerät
Size	Gesamtgröße
Used	belegt
Avail	frei
Use%	prozentuale Nutzung
Mounted on	Mountpoint

Merksatz:

```
df zeigt,
wie voll Dateisysteme sind.
```

---

df -h

Die Option -h bedeutet:

```
human-readable
```

Also:

```
menschenlesbar
```

Beispiel:

```
df -h
```

zeigt Größen in:

```
K

M

G

T
```

statt nur in Blöcken oder Bytes.

Merksatz:

```
df -h ist Standard zur Speicherplatzprüfung.
```

---

du

du zeigt, wie viel Speicher Dateien oder Verzeichnisse belegen.

Beispiele:

```
du -sh /var/log

du -sh /home/felix

du -h /var/log
```

Bedeutung:

Befehl	Bedeutung
du -sh ordner	Gesamtgröße eines Ordners
du -h ordner	Größen rekursiv anzeigen
du -sh *	Größen im aktuellen Verzeichnis zusammenfassen

Merksatz:

```
df zeigt Dateisysteme.
du zeigt Verzeichnisgrößen.
```

---

df und du unterscheiden

Befehl	Zeigt
df	freien und belegten Platz von Dateisystemen
du	Speicherverbrauch von Dateien und Ordnern

Beispiel:

```
df -h
```

zeigt, ob /var voll ist.

```
du -sh /var/log
```

zeigt, wie groß der Logordner ist.

Merksatz:

df für Dateisystem.
du für Ordner.

Speicherplatz voll

Wenn ein Dateisystem voll ist, können viele Probleme entstehen.

Mögliche Folgen:

Dienste starten nicht

Logs können nicht geschrieben werden

Datenbanken stoppen oder beschädigen Transaktionen

Updates schlagen fehl

Benutzer können keine Dateien speichern

Backups schlagen fehl

temporäre Dateien können nicht erstellt werden

System wird instabil

Merksatz:

Volle Dateisysteme verursachen viele scheinbar unterschiedliche Fehler.

Typische volle Verzeichnisse

Häufige Ursachen für vollen Speicher:

Verzeichnis	Möglicher Grund
/var/log	zu viele oder zu große Logs
/var/lib	Datenbanken, Docker, Anwendungen
/tmp	temporäre Dateien

Verzeichnis	Möglicher Grund
/home	Benutzerdateien
/var/cache	Paketcache oder Anwendungscache
/var/spool	Warteschlangen, Mail, Druck
/backup	alte Sicherungen

Merksatz:

```
Bei vollem Speicher zuerst /var,
Logs,
Cache
und Anwendungsdaten prüfen.
```

Inodes

Neben Speicherplatz gibt es Inodes.

Ein Inode ist eine Verwaltungsstruktur für Dateien.

Ein Dateisystem kann voll sein, auch wenn noch Speicherplatz frei ist, wenn alle Inodes verbraucht sind.

Prüfen:

```
df -i
```

Typische Ursache:

```
sehr viele kleine Dateien
```

Beispiele:

```
Cache-Dateien

Session-Dateien

Mailspool
```

```
temporäre Dateien
```

Merksatz:

```
Speicher kann auch durch zu viele Dateien knapp werden.
```

---

df -i

df -i zeigt Inode-Nutzung.

Beispiel:

```
df -i
```

Wichtige Spalten:

```
Inodes

IUsed

IFree

IUse%
```

Wenn IUse% bei 100 Prozent liegt, können keine neuen Dateien mehr angelegt werden, auch wenn noch Speicherplatz frei ist.

Merksatz:

```
df -i prüft,
ob zu viele Dateien existieren.
```

/etc/fstab

Die Datei:

```
/etc/fstab
```

legt fest, welche Dateisysteme beim Systemstart automatisch eingebunden werden.

fstab steht für:

```
file system table
```

Typische Inhalte:

```
Gerät oder UUID

Mountpoint

Dateisystemtyp

Mountoptionen

Dump

fsck-Reihenfolge
```

Beispielzeile:

```
UUID=abcd-1234 /mnt/backup ext4 defaults 0 2
```

Merksatz:

```
/etc/fstab regelt automatische Mounts beim Start.
```

Warum UUIDs in fstab sinnvoll sind

Gerätenamen wie /dev/sdb1 können sich ändern, zum Beispiel wenn Datenträger in anderer Reihenfolge erkannt werden.

UUIDs sind eindeutige Kennungen von Dateisystemen.

Anzeigen:

```
lsblk -f
```

oder:

```
blkid
```

Vorteil:

```
Mounts bleiben stabiler,
auch wenn sich Gerätenamen ändern.
```

Merksatz:

```
UUID ist zuverlässiger als /dev/sdX für dauerhafte Mounts.
```

---

fstab-Fehler

Fehler in /etc/fstab können dazu führen, dass das System beim Start Probleme bekommt oder ein Dateisystem nicht eingehängt wird.

Typische Fehler:

```
falsche UUID

falscher Mountpoint

Mountpoint existiert nicht

falscher Dateisystemtyp

falsche Optionen

Netzwerkfreigabe beim Boot nicht erreichbar
```

Sicher prüfen:

```
mount -a
```

Bedeutung:

```
mount -a versucht,
alle fstab-Einträge einzuhängen.
```

Merksatz:

```
fstab-Änderungen immer testen,
bevor man neu startet.
```

---

mount -a

Der Befehl:

```
mount -a
```

liest /etc/fstab und versucht, alle noch nicht gemounteten Einträge einzuhängen.

Nutzen:

```
fstab testen

Fehler vor Neustart erkennen
```

Wichtig:

```
Bei Fehlern Ausgabe genau lesen.
```

Merksatz:

```
mount -a prüft fstab praktisch.
```

Mountoptionen

Mountoptionen steuern, wie ein Dateisystem eingebunden wird.

Beispiele:

Option	Bedeutung

defaults	Standardoptionen
ro	read-only, nur lesbar
rw	read-write, lesbar und beschreibbar
noexec	keine Programme ausführen
nosuid	SetUID ignorieren
nodev	Gerätedateien ignorieren
noatime	Zugriffszeit nicht ständig aktualisieren
user	Benutzer dürfen mounten

Merksatz:

```
Mountoptionen beeinflussen Sicherheit und Verhalten.
```

---

read-only Mount

Ein Dateisystem kann nur lesbar eingebunden sein.

Anzeige möglich über:

```
mount
```

oder:

```
findmnt
```

Typische Ursache:

```
bewusst so konfiguriert

Dateisystemfehler

Schutzmodus

Rettungsmodus
```

Folge:

```
Schreiben ist nicht möglich.
```

Merksatz:

```
ro bedeutet read-only.
rw bedeutet read-write.
```

---

findmnt

findmnt zeigt Mounts übersichtlich an.

Befehl:

```
findmnt
```

Bestimmten Mountpoint prüfen:

```
findmnt /mnt/backup
```

Merksatz:

```
findmnt zeigt eingebundene Dateisysteme strukturiert.
```

---

blkid

blkid zeigt Informationen zu Blockgeräten.

Befehl:

```
blkid
```

Typische Informationen:

```
UUID

Dateisystemtyp
```

```
Label
```

Merksatz:

```
blkid hilft,
UUIDs und Dateisysteme zu erkennen.
```

Dateisystemprüfung

Dateisysteme können geprüft und repariert werden.

Typisches Werkzeug:

```
fsck
```

Wichtig:

```
Dateisysteme sollten in der Regel nicht im eingehängten,
aktiv genutzten Zustand repariert werden.
```

Risiko:

```
Datenverlust

weitere Beschädigung

inkonsistenter Zustand
```

Merksatz:

```
fsck vorsichtig verwenden,
besonders auf produktiven Systemen.
```

Swap

Swap ist Auslagerungsspeicher.

Wenn RAM knapp wird, kann Linux Speicherbereiche auf Datenträger auslagern.

Swap kann sein:

```
Swap-Partition

Swap-Datei
```

Prüfen:

```
swapon --show

free -h
```

Wichtig:

```
Swap ersetzt keinen ausreichenden RAM.
Sehr starke Swap-Nutzung kann das System langsam machen.
```

Merksatz:

```
Swap hilft bei Speicherdruck,
ist aber langsamer als RAM.
```

---

free -h

free zeigt Arbeitsspeicher und Swap.

Befehl:

```
free -h
```

Typische Werte:

```
total

used

free
```

```
available

swap
```

Merksatz:

```
free -h zeigt RAM und Swap.
```

Speicher und Container

Container nutzen häufig:

```
Images

Container-Dateisysteme

Volumes

Bind Mounts

Logs

Caches
```

Typische Speicherorte je nach System:

```
/var/lib/docker

/var/lib/containerd
```

Wichtig:

```
Container-Logs,
Images
und Volumes können viel Speicher belegen.
```

Merksatz:

Container können Speicher durch Images,
Volumes
und Logs verbrauchen.

Volumes und Bind Mounts

Container können Daten außerhalb des Containers speichern.

Volume:

vom Container-System verwalteter Speicher

Bind Mount:

bestimmter Host-Ordner wird in Container eingebunden

Beispielhafte Idee:

Host-Ordner:
/srv/appdata

im Container:
/data

Vorteil:

Daten bleiben erhalten,
auch wenn der Container neu erstellt wird.

Merksatz:

Containerdaten sollten nicht nur im Container-Dateisystem liegen.

Netzwerkspeicher

Linux kann Netzwerkspeicher einbinden.

Beispiele:

```
NFS

SMB/CIFS

SSHFS
```

Typische Nutzung:

```
Backups

gemeinsame Daten

NAS-Freigaben

zentrale Projektordner
```

Wichtig:

```
Netzwerk-Mounts hängen von Netzwerk,
DNS,
Berechtigungen
und Serververfügbarkeit ab.
```

Merksatz:

```
Netzwerkspeicher braucht Netzwerk und Rechte.
```

NFS

NFS steht für:

```
Network File System
```

Typischer Einsatz:

Linux/Unix-Netzwerkfreigaben

Vorteile:

gut in Linux-Umgebungen integriert

geeignet für Server-zu-Server-Freigaben

Wichtig:

Rechte,
UID/GID
und Export-Regeln müssen passen.

Merksatz:

NFS ist typisch für Linux/Unix-Freigaben.

SMB/CIFS

SMB wird häufig für Windows-Freigaben genutzt, kann aber auch unter Linux eingebunden werden.

Typische Nutzung:

Windows-Freigaben

NAS-Freigaben

gemischte Umgebungen

Wichtig:

Benutzer,
Passwort,
Domain,
Rechte
und Mountoptionen müssen passen.

Merksatz:

> SMB/CIFS ist typisch für Windows-
> und NAS-Freigaben.

Backup-Speicher

Backup-Speicher sollte besonders geschützt werden.

Wichtig:

> getrennte Rechte
>
> keine normalen Benutzer mit Löschrechten
>
> Monitoring
>
> genügend Speicherplatz
>
> Retention
>
> Restore-Test
>
> Schutz vor Ransomware

Merksatz:

> Backup-Speicher ist sicherheitskritisch.

Typische Speicherbefehle

Befehl	Zweck
lsblk	Datenträger und Partitionen anzeigen
lsblk -f	Dateisysteme und UUIDs anzeigen
blkid	UUIDs und Dateisystemtypen anzeigen
df -h	Speicherplatz von Dateisystemen anzeigen
df -i	Inode-Nutzung anzeigen

Befehl	Zweck
du -sh ordner	Größe eines Ordners anzeigen
mount	Mounts anzeigen oder einhängen
umount	Dateisystem aushängen
findmnt	Mounts strukturiert anzeigen
free -h	RAM und Swap anzeigen
swapon --show	Swap anzeigen

Typische Fehlerbilder

Fehlerbild	Wahrscheinliche Ursache
No space left on device	Speicherplatz oder Inodes voll
Read-only file system	Dateisystem nur lesbar eingebunden oder Fehler
target is busy	Mount wird noch verwendet
mount: wrong fs type	falscher Dateisystemtyp oder fehlender Treiber
Mount nach Neustart fehlt	fstab fehlt oder fehlerhaft
System startet mit Mountfehler	fstab-Fehler
Dienst kann nicht schreiben	Rechte, voller Speicher oder read-only
Backup schlägt fehl	Ziel voll, nicht gemountet oder Rechteproblem
Daten scheinbar verschwunden	Mount nicht aktiv oder anderer Mount überlagert Verzeichnis
Container belegt viel Speicher	Images, Volumes oder Logs

No space left on device

Diese Meldung bedeutet:

```
Auf dem Ziel-Dateisystem kann nicht geschrieben werden.
```

Mögliche Ursachen:

```
Speicherplatz voll

Inodes voll
```

Quota erreicht

Prüfen:

```
df -h

df -i

du -sh pfad
```

Merksatz:

```
Bei No space left on device Speicherplatz und Inodes prüfen.
```

Read-only file system

Diese Meldung bedeutet:

```
Das Dateisystem ist nur lesbar eingebunden.
```

Mögliche Ursachen:

```
bewusst read-only gemountet

Dateisystemfehler

Schutzmodus nach Fehler

Rettungssystem
```

Prüfen:

```
mount

findmnt

Logs
```

Merksatz:

```
read-only verhindert Schreibzugriffe.
```

Mount überlagert vorhandene Daten

Wenn ein Dateisystem auf ein Verzeichnis gemountet wird, verdeckt es die bisherigen Inhalte dieses Verzeichnisses.

Beispiel:

```
In /mnt/data liegen Dateien.

Dann wird ein anderes Dateisystem auf /mnt/data gemountet.

Nun sieht man die Inhalte des gemounteten Dateisystems.
```

Die ursprünglichen Dateien sind nicht weg, aber durch den Mount verdeckt.

Merksatz:

```
Mounts können vorhandene Verzeichnisinhalte verdecken.
```

Sicheres Vorgehen bei fstab-Änderungen

Vor Änderung:

```
aktuelle fstab sichern

cp /etc/fstab /etc/fstab.bak
```

Änderung durchführen:

```
nano /etc/fstab
```

Testen:

```
mount -a
```

Prüfen:

```
findmnt

df -h
```

Wichtig:

```
Erst testen,
dann neu starten.
```

Merksatz:

```
fstab nie ungetestet ändern.
```

Sichere Arbeitsweise bei Speicherproblemen

Bei Speicherproblemen:

```
1. Fehler genau lesen

2. df -h prüfen

3. df -i prüfen

4. du -sh für große Ordner nutzen

5. Logs prüfen

6. Mountstatus prüfen

7. Rechte prüfen

8. Dienste erst danach neu starten
```

Nicht sofort:

wahllos Dateien löschen

Logs blind entfernen

Datenbanken im laufenden Betrieb löschen

Backup-Verzeichnisse ohne Prüfung leeren

Merksatz:

Speicherprobleme erst messen,
dann gezielt bereinigen.

Typische Prüfungsfragen

Frage	Kurzantwort
Was ist ein Datenträger?	physisches oder virtuelles Speichermedium
Was ist eine Partition?	Bereich auf einem Datenträger
Was ist ein Dateisystem?	Organisationsform für Dateien
Was ist ein Mountpoint?	Einhängepunkt im Verzeichnisbaum
Was macht mount?	Dateisystem einhängen oder Mounts anzeigen
Was macht umount?	Dateisystem aushängen
Was zeigt lsblk?	Datenträger und Partitionen
Was zeigt df -h?	freien und belegten Speicherplatz
Was zeigt du -sh?	Größe eines Ordners
Was zeigt df -i?	Inode-Nutzung
Was macht /etc/fstab?	automatische Mounts beim Systemstart
Warum UUID statt /dev/sdb1?	stabiler bei wechselnder Geräteerkennung
Was ist Swap?	Auslagerungsspeicher
Was bedeutet read-only?	nur lesbar eingebunden

Typische Prüfungsfallen

Falle	Richtig denken
Linux mit C: und D: erklären	Linux nutzt Mountpoints

Falle	Richtig denken
df und du verwechseln	df Dateisystem, du Ordner
Speicher frei, aber keine Datei möglich	Inodes prüfen
fstab ungetestet ändern	immer mount -a testen
/dev/sdb1 fest erwarten	Gerätenamen können sich ändern
UUID ignorieren	für dauerhafte Mounts besser
umount falsch als unmount schreiben	Befehl heißt umount
Mountpoint nicht vorhanden	Verzeichnis vorher erstellen
Mount verdeckt vorhandene Daten	Daten sind überlagert
Swap mit RAM gleichsetzen	Swap ist langsamer Auslagerungsspeicher
Containerdaten im Container lassen	Volumes oder Bind Mounts nutzen
Backup-Speicher normal freigeben	Sicherheitsrisiko

IHK-sichere Kurzformulierung

Linux verwendet keine Laufwerksbuchstaben, sondern bindet Dateisysteme über Mountpoints in den gemeinsamen Verzeichnisbaum ein. Ein Datenträger kann in Partitionen aufgeteilt werden, auf denen Dateisysteme wie ext4, XFS oder Btrfs erstellt werden. Mit lsblk werden Datenträger und Partitionen angezeigt, df -h zeigt belegten und freien Speicherplatz von Dateisystemen und du -sh zeigt die Größe einzelner Verzeichnisse. Die Datei /etc/fstab legt fest, welche Dateisysteme beim Systemstart automatisch eingebunden werden. Für dauerhafte Mounts sind UUIDs oft zuverlässiger als Gerätenamen wie /dev/sdb1, weil sich Gerätenamen ändern können. Bei Speicherproblemen sollten Speicherplatz, Inodes, Mountstatus, Rechte und Logs geprüft werden.

Merksätze

```
Linux nutzt Mountpoints statt Laufwerksbuchstaben.

Datenträger erscheinen oft unter /dev.

Partition = Bereich auf einem Datenträger.

Dateisystem = Organisationsform für Dateien.

Mountpoint = Einhängepunkt.

mount hängt ein.
```

umount hängt aus.

Der Befehl heißt umount,
nicht unmount.

lsblk zeigt Datenträger und Partitionen.

lsblk -f zeigt Dateisysteme und UUIDs.

df -h zeigt Speicherplatz von Dateisystemen.

du -sh zeigt Größe eines Ordners.

df -i zeigt Inode-Nutzung.

No space left on device:
Speicherplatz oder Inodes prüfen.

/etc/fstab regelt automatische Mounts.

fstab-Änderungen mit mount -a testen.

UUIDs sind stabiler als /dev/sdX.

ro bedeutet read-only.

rw bedeutet read-write.

findmnt zeigt Mounts übersichtlich.

Swap ist Auslagerungsspeicher.

Swap ist langsamer als RAM.

Netzwerk-Mounts brauchen Netzwerk,
DNS
und Rechte.

Containerdaten brauchen Volumes oder Bind Mounts.

Backup-Speicher besonders schützen.

Mounts können vorhandene Verzeichnisinhalte verdecken.

Erst messen,
dann löschen.