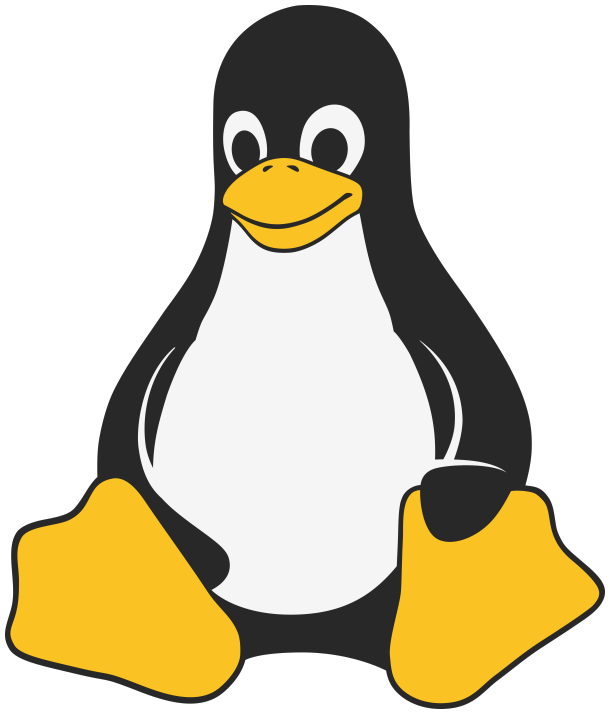
Festplatten-Verwaltung



Inhaltsverzeichnis

- Neue Festplatte in `deb-srv`: `/dev/sdb`
- `/dev/sdb` partitionieren
- `/dev/sdb1` formatieren mit `ext4`-Filesystem
- `/dev/sdb1`-Dateisystem prüfen mit `fsck`
- `/dev/sdb1` manuell einhängen
- `/dev/sdb2` formatieren mit `btrfs`-Filesystem
- `/dev/sdb2`-Dateisystem prüfen mit `fsck`
- `/dev/sdb2` manuell einhängen

- Automatisches Einhängen: `/etc/fstab`
- Einhängen mit `sudo mount -a`
- Einhängen mit UUID statt Gerätedatei
- Verwendung der neuen Dateisysteme

Hinweis

Die Konzepte, die die Grundlage für die Verwaltung von Festplatten, SSDs und anderen Speichermedien bilden, wurden bereits in Chap05/L01-Dateisystem behandelt.

Dieses Grundwissen wird in diesem Kapitel vorausgesetzt. Hier geht es um die praktische Anwendung dieser Konzepte in der Administration von Speichermedien, die in ein Linux-System eingebunden werden.

Zum Inhaltsverzeichnis ...

Lokale physische Dateisysteme

Mit dem Kommando `df` (disk free) wird der Speicherplatz auf den Dateisystemen angezeigt, die im Dateibaum eingehängt sind.

Um nur die physischen Dateisysteme anzuzeigen, verwenden wir auf `debian` und `dev-srv` den Alias `dfp`.

Zum Inhaltsverzeichnis ...

Alias `dfp` für `df -T -x tmpfs -x devtmpfs`

```
hermann@debian:~$ alias dfp='df -T -x tmpfs -x devtmpfs'
hermann@debian:~$ dfp
Dateisystem    Typ  1K-Blöcke Benutzt Verfügbar Verw% Eingehängt auf
/dev/sda2      ext4  18964304 5078268  12897364   29% /
/dev/sda1      vfat    523244    5984    517260    2% /boot/efi
```

Dieser Alias nimmt die Typ-Spalte in die Ausgabe von `df` auf und filtert die virtuellen Dateisysteme `tmpfs` und `devtmpfs` aus. `dfp` zeigt also nur die physischen Dateisysteme an.

Definieren Sie diesen Alias in ~/.bash_aliases (auf `debian` und `deb-srv`), sodass er in jeder Shell-Sitzung verfügbar ist.

Lokale Dateisysteme auf `debian`

Bei der Installation von Debian haben uns entschieden, alles auf einer einzigen Partition zu installieren. Dadurch wurde die Platte `/dev/sda` in drei Partitionen aufgeteilt. Zwei davon sind eingehängt und erscheinen in der Ausgabe von `df`.

Die dritte Partition `/dev/sda3` ist die **Swap-Partition**. Sie wird nicht eingehängt und erscheint deshalb nicht in der Ausgabe von `df`. Sie wird jedoch von `swapon` angezeigt.

```
hermann@debian:~$ dfp
Dateisystem    Typ  1K-Blöcke Benutzt Verfügbar Verw% Eingehängt auf
/dev/sda2      ext4  18964304 5078268  12897364   29% /
/dev/sda1      vfat    523244    5984    517260    2% /boot/efi
```

```
hermann@debian:~$ sudo swapon -s
Filename        Type            Size            Used            Priority
/dev/sda3       partition       999420          0               -2
```

Lokale Dateisysteme auf `dev-srv`

Bei der Installation des Debian-Servers `dev-srv` haben wir uns entschieden, für die Dateien unter `/var`, `/home` und `/tmp` separate Partitionen zu erstellen. Es erscheinen deshalb drei zusätzliche Partitionen in der Ausgabe von `df`.

```
hermann@deb-srv:~$ dfp
Dateisystem    Typ  1K-Blöcke Benutzt Verfügbar Verw% Eingehängt auf
/dev/sda2      ext4   4074632 1259760   2587340   33% /
/dev/sda1      vfat    523244    5984    517260    2% /boot/efi
/dev/sda3      ext4   1667512  298392   1266108   20% /var
/dev/sda6      ext4  12800688     228  12128412    1% /home
/dev/sda5      ext4    341983       8    319089    1% /tmp
```

```
hermann@deb-srv:~$ sudo swapon -s
Filename        Type            Size            Used            Priority
/dev/sda4       partition       1000444         0               -2
```

Namensschema für Speichermedien

Speichermedien sind "blockorientierte Gerätedateien" im Verzeichnis `/dev`.

Sie haben Namen wie `/dev/sda` für die erste Festplatte, `/dev/sdb` für die zweite Festplatte, `/dev/sdc` für die dritte Festplatte usw.

Die Partitionen auf der ersten Festplatte haben Namen wie `/dev/sda1`, `/dev/sda2`, usw. Auf der zweiten Festplatte heißen die Partitionen `/dev/sdb1`, `/dev/sdb2`, usw.

Zum Inhaltsverzeichnis ...

Blockgerätedateien anzeigen

- Mit `ls -l /dev/sd*` lassen sich die blockorientierten Gerätedateien anzeigen, die mit `/dev/sd` beginnen.
- Das Kommando `lsblk` zeigt eine Baumstruktur der Blockgeräte an. Es zeigt die Geräte und Partitionen, die im System vorhanden sind, und wie sie miteinander verbunden sind.

```
hermann@debian:~$ ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0  9. Dez 19:09 /dev/sda
brw-rw---- 1 root disk 8, 1  9. Dez 19:07 /dev/sda1
brw-rw---- 1 root disk 8, 2  9. Dez 19:09 /dev/sda2
brw-rw---- 1 root disk 8, 3  9. Dez 19:09 /dev/sda3
```

```
hermann@debian:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   20G  0 disk
├─sda1   8:1    0  512M  0 part /boot/efi
├─sda2   8:2    0 18,5G  0 part /
└─sda3   8:3    0  976M  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

Festplatten-Verwaltung - Überblick

Es geht in diesem Foliensatz um die Verwaltung von Festplatten (und anderen Speichermedien) und deren Partitionen.

Um ein lokales Speichermedium zu verwenden,

- muss es partitioniert werden.
- Auf jeder Partition muss ein Dateisystem erstellt werden. D.h. die Partition wird formatiert.
- Die Dateisysteme der Partitionen müssen in den Dateibaum des Linux-Systems eingehängt (gemountet) werden.

Partitionierung anzeigen mit `fdisk -l`

Das Kommando `fdisk -l` gibt ausführliche Informationen die Festplatte selbst aus, z.B. die Größe der Festplatte, die Anzahl der Sektoren.

Es zeigt außerdem sektorgenaue Angaben über die Lage und Größe der Partitionen auf der Festplatte.

Ohne die Option `-l` man das `fdisk`-Kommando im interaktiven Modus zur Partitionierung von Festplatten verwenden.

```
hermann@debian:~$ sudo fdisk -l
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: Virtual Disk
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: gpt
Disk identifier: F57AAC3B-99CE-49EE-B8B6-12504834CA8C

Device         Start      End  Sectors  Size Type
/dev/sda1       2048  1050623  1048576  512M EFI System
/dev/sda2    1050624 39942143 38891520 18,5G Linux filesystem
/dev/sda3   39942144 41940991  1998848  976M Linux swap
```

```
hermann@deb-srv:~$ sudo fdisk -l
[sudo] Passwort für hermann:
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: Virtual Disk
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: gpt
Disk identifier: B7F05CCD-9F42-45EC-8065-43DEDD879DB5

Device         Start      End  Sectors  Size Type
/dev/sda1       2048  1050623  1048576  512M EFI System
/dev/sda2    1050624  9496575  8445952    4G Linux filesystem
/dev/sda3    9496576 12961791  3465216  1,7G Linux filesystem
/dev/sda4   12961792 14962687  2000896  977M Linux swap
/dev/sda5   14962688 15714303   751616  367M Linux filesystem
/dev/sda6   15714304 41940991 26226688 12,5G Linux filesystem
```

Weiteres Vorgehen

An der ersten Festplatte wollen wir nicht herumexperimentieren, um unsere Linux-Installation nicht zu gefährden.

- Deshalb erstellen wir eine neue virtuelle Festplatte im Hyper-V-Manager und fügen sie dem Debian-Server `dev-srv` hinzu.
- Wir prüfen, ob die neue Festplatte jetzt in unserem Linux-System `deb-srv` als `/dev/sdb` erscheint.
- Wir partitionieren sie und erstellen mit `fdisk` zwei Partitionen: `/dev/sdb1` und `/dev/sdb2`.

- Wir formatieren `/dev/sdb1` mit `ext4`.
- Wir formatieren `/dev/sdb2` mit `btrfs`.
- Wir hängen beide Partitionen manuell in den Dateibaum ein und wieder aus.
- Wir tragen beide Partitionen in die Datei `/etc/fstab` ein, damit sie beim Systemstart automatisch eingehängt werden.
- Wir verwenden für das Einhängen der Partitionen in `/etc/fstab` UUIDs statt der Gerätedateien.

Neue virtuelle Festplatte in die VM `dev-srv` "einbauen"

Wie eine neue virtuelle Festplatte im Hyper-V-Manager erstellt und einer virtuellen Maschine hinzugefügt wird, wird in folgendem Foliensatz beschrieben:

- *L02a-Virtuelle-Festplatte-in-virtuelle-Maschine-einbauen.md*

Neue Festplatte in `deb-srv`: `/dev/sdb`

Vor dem "Einbau" der neuen Festplatte existierte nur die Festplatte `/dev/sda` im Debian-Server `dev-srv`. Nun erwarten wir eine neue Gerätedatei `/dev/sdb`. Da die Platte noch nicht partitioniert ist, existieren noch keine Partitionen `/dev/sdb1` und `/dev/sdb2`.

```
hermann@deb-srv:~$ ls -l /dev/sdb*
brw-rw---- 1 root disk 8, 16  9. Dez 20:30 /dev/sdb
```

```
hermann@deb-srv:~$ lsblk /dev/sdb
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sdb    8:16   0  10G  0 disk
```

```
hermann@deb-srv:~$ sudo fdisk -l /dev/sdb
[sudo] Passwort für hermann:
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

`/dev/sdb` partitionieren

Das `fdisk`-Kommando (ohne Option `-l`) wird im interaktiven Modus verwendet, um Festplatten zu partitionieren.

Dabei werden Kommandos bestehend aus einem einzelnen Buchstaben eingegeben, um das `fdisk`-Programm zu steuern. Hilfe bietet das Kommando `m`. Mit dem Kommando `p` wird die aktuelle Partitionstabelle angezeigt. Erst mit dem Kommando `w` werden die Änderungen auf die Festplatte geschrieben.

Zum Inhaltsverzeichnis ...

`fdisk` wird mit `sudo` gestartet

```
hermann@deb-srv:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.38.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS (MBR) disklabel with disk identifier 0x34ed3ea1.
```

`fdisk`-Kommando `g` erstellt eine neue GPT-Partitionstabelle

```
Command (m for help): g
Created a new GPT disklabel (GUID: D0049A36-D8EC-5648-B9CB-5096C8843022).
The device contains 'dos' signature and it will be removed by a write command.
See fdisk(8) man page and --wipe option for more details.
```

GPT (GUID Partition Table) ist der moderne Nachfolger des MBR (Master Boot Record) für Partitionstabellen. GPT unterstützt Partitionen mit einer Größe von mehr als 2 TB und mehr als vier primären Partitionen. GPT ist der Standard für neue Festplatten und besonders für UEFI-Systeme und große Festplatten oder SSDs zu empfehlen.

`fdisk`-Kommando `n` legt eine neue Partition an

Wir erstellen eine primäre Partition mit einer Größe von 5 GB.

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-20971519, default 20971519): +5G

Created a new partition 1 of type 'Linux' and of size 5 GiB.
```

`fdisk`-Kommando `p` zeigt die aktuelle Partitionstabelle

(Die Partitionstabelle ist noch nicht auf die Platte geschrieben.)

```
Command (m for help): p
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: Virtual Disk
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: 0x34ed3ea1

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1        2048 10487807 10485760   5G 83 Linux
```

`fdisk`-Kommando `n` legt eine zweite Partition an

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2):
First sector (10487808-20971519, default 10487808):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (10487808-20971519, default 20971519):

Created a new partition 2 of type 'Linux' and of size 5 GiB.
```

`fdisk`-Kommando `p` zeigt die aktuelle Partitionstabelle

(Die Partitionstabelle ist noch nicht auf die Platte geschrieben.)

```
Command (m for help): p
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: Virtual Disk
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: gpt
Disk identifier: D0049A36-D8EC-5648-B9CB-5096C8843022

Device         Start      End  Sectors Size Type
/dev/sdb1       2048 10487807 10485760   5G Linux filesystem
/dev/sdb2   10487808 20969471 10481664   5G Linux filesystem
```

`fdisk`-Kommando `w` schreibt die Partitionstabelle auf die Platte

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Durch die Partitionierung von `/dev/sdb` wurden die neuen Gerätedateien `/dev/sdb1` und `/dev/sdb2` erstellt.

Partitionierung prüfen

```
hermann@deb-srv:~$ ls -l /dev/sdb*
brw-rw---- 1 root disk 8, 16 10. Dez 15:05 /dev/sdb
brw-rw---- 1 root disk 8, 17 10. Dez 15:05 /dev/sdb1
brw-rw---- 1 root disk 8, 18 10. Dez 15:05 /dev/sdb2
```

```
hermann@deb-srv:~$ lsblk /dev/sdb
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sdb      8:16   0  10G  0 disk
├─sdb1   8:17   0   5G  0 part
└─sdb2   8:18   0   5G  0 part
```

```
hermann@deb-srv:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: Virtual Disk
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: gpt
Disk identifier: D0049A36-D8EC-5648-B9CB-5096C8843022

Device         Start      End  Sectors Size Type
/dev/sdb1       2048 10487807 10485760   5G Linux filesystem
/dev/sdb2   10487808 20969471 10481664   5G Linux filesystem
```

`fdisk` zeigt den Partitionstyp `Linux filesystem` an. Das sagt nichts über den Typ des Dateisystems aus, das bei der Formatierung der Partitionen erstellt wird.

`/dev/sdb1` formatieren mit `ext4`-Filesystem

Das Kommando `mkfs` (make file system) wird verwendet, um ein Dateisystem auf einer Partition zu erstellen, bzw. eine Partition mit einem Dateisystem zu formatieren. Mit der Option `-t <typ>` ist der Dateisystemtyp anzugeben.

Viele weitere dateisystemspezifische Optionen sind verfügbar.

`mkfs -t <type> <device>` ist das generische Kommando, z.B. `mkfs -t ext4 <device>`. Dieses ruft das spezifische Kommando für das Dateisystem auf, z.B. `mkfs.ext4 <device>`.

Man kann auch direkt das dateisystemspezifische Kommando verwenden, z.B. `mkfs.ext4 <device>`.

Es macht keinen Unterschied, ob wir das generische oder das spezifische Kommando verwenden.

```
hermann@deb-srv:~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.47.0 (5-Feb-2023)
Discarding device blocks: done
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: 1d396ff6-cda5-4f78-8324-1a118cbdb574
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Das Dateisystem `ext4` wurde auf `/dev/sdb1` erstellt. Die UUID des Dateisystems ist `1d396ff6-cda5-4f78-8324-1a118cbdb574`. Wir können das Dateisystem jetzt in den Dateibaum einhängen.

`/dev/sdb1` -Dateisystem prüfen

Dateisysteme können inkonsistent werden, z.B. durch einen unerwarteten Systemabsturz. Das Kommando `fsck` (file system check) prüft und repariert Dateisysteme.

```
hermann@deb-srv:~$ sudo fsck /dev/sdb1
fsck from util-linux 2.38.1
e2fsck 1.47.0 (5-Feb-2023)
/dev/sdb1: clean, 11/327680 files, 42078/1310720 blocks
```

Der Typ des Dateisystems kann mit der Option `-t <typ>` angegeben werden. Dies ist jedoch nicht erforderlich, da `fsck` den Typ des Dateisystems automatisch erkennt.

`/dev/sdb1` manuell einhängen

Um ein Dateisystem Dateisystem zu verwenden, muss es in den Linux-Datenbaum eingehängt werden. Das Kommando `mount` wird verwendet, um ein Dateisystem einzuhängen. Als Montierungspunkt ist ein leeres Verzeichnis anzugeben.

```
hermann@deb-srv:~$ sudo mkdir /data1
hermann@deb-srv:~$ sudo mount /dev/sdb1 /data1
```

Auch hier kann die Angabe des Dateisystemtyps mit der Option `-t <typ>` unterbleiben, da `mount` den Typ aus der Formatierung der Partition auslesen kann.

Prüfen, ob `/dev/sdb1` eingehängt ist

- mit `mount

```
hermann@deb-srv:~$ mount | grep sdb1
/dev/sdb1 on /data1 type ext4 (rw,relatime)
```

- mit `df`

```
hermann@deb-srv:~$ df -T | grep sdb1
/dev/sdb1      ext4       5074592      24   4796040    1% /data1
```

`/dev/sdb1` wieder aushängen

Ein eingehängtes Dateisystem wird mit dem Kommando `umount` wieder ausgehängt.
Als Argument ist die Gerätedatei oder der Montierpunkt anzugeben.

- `umount` mit Gerätedatei

```
hermann@deb-srv:~$ mount | grep sdb1 # mounted? yes
/dev/sdb1 on /data1 type ext4 (rw,relatime)
hermann@deb-srv:~$ sudo umount /dev/sdb1 # umount device
hermann@deb-srv:~$ mount | grep sdb1 # mounted? no
```

- `umount` mit Montierpunkt

```
hermann@deb-srv:~$ sudo mount /dev/sdb1 /data1 # mount again
hermann@deb-srv:~$ mount | grep sdb1 # mounted? yes
/dev/sdb1 on /data1 type ext4 (rw,relatime)
hermann@deb-srv:~$ sudo umount /data1 # umount mountpoint
hermann@deb-srv:~$ mount | grep sdb1 # mounted? no
```

`/dev/sdb2` formatieren mit `btrfs`-Filesystem

Analog zur Formatierung von `/dev/sdb1` mit `ext4` wird `/dev/sdb2` mit `btrfs` formatiert.

```
hermann@deb-srv:~$ sudo mkfs -t btrfs /dev/sdb2
mkfs: failed to execute mkfs.btrfs: Datei oder Verzeichnis nicht gefunden
hermann@deb-srv:~$ sudo mkfs.btrfs /dev/sdb2
sudo: mkfs.btrfs: Befehl nicht gefunden
hermann@deb-srv:~$ sudo apt install btrfs-progs # install btrfs support
Paketlisten werden gelesen… Fertig
...
hermann@deb-srv:~$ sudo which mkfs.btrfs
/usr/sbin/mkfs.btrfs
```


Zum Inhaltsverzeichnis …

```
hermann@deb-srv:~$ sudo mkfs -t btrfs /dev/sdb2
btrfs-progs v6.2
See http://btrfs.wiki.kernel.org for more information.

...

Label:              (null)
UUID:               2e1968f6-fa95-4c89-aae9-27089aefc4da
Node size:          16384
Sector size:        4096
Filesystem size:    5.00GiB
...
Checksum:           crc32c
Number of devices:  1
Devices:
   ID        SIZE  PATH
    1     5.00GiB  /dev/sdb2
```

`/dev/sdb2` -Dateisystem prüfen

Wir prüfen das `btrfs`-Dateisystem auf `/dev/sdb2` zunächst wie bei `ext4` mit `fsck`, erhalten jedoch eine Fehlermeldung.

```
hermann@deb-srv:~$ sudo fsck /dev/sdb2 # try fsck for btrfs
fsck from util-linux 2.38.1
If you wish to check the consistency of a BTRFS filesystem or
repair a damaged filesystem, see btrfs(8) subcommand 'check'.
hermann@deb-srv:~$ sudo fsck.btrfs /dev/sdb2 # try fsck.btrfs
If you wish to check the consistency of a BTRFS filesystem or
repair a damaged filesystem, see btrfs(8) subcommand 'check'.
```

`fsck` kann das `btrfs`-Dateisystem nicht prüfen. Dafür gibt es das Kommando `btrfs check`.

```
hermann@deb-srv:~$ sudo btrfs check /dev/sdb2
Opening filesystem to check...
Checking filesystem on /dev/sdb2
UUID: 2e1968f6-fa95-4c89-aae9-27089aefc4da
[1/7] checking root items
[2/7] checking extents
[3/7] checking free space tree
[4/7] checking fs roots
[5/7] checking only csums items (without verifying data)
[6/7] checking root refs
[7/7] checking quota groups skipped (not enabled on this FS)
found 147456 bytes used, no error found
total csum bytes: 0
total tree bytes: 147456
total fs tree bytes: 32768
total extent tree bytes: 16384
btree space waste bytes: 140595
file data blocks allocated: 0
 referenced 0
```

`/dev/sdb2` manuell einhängen

Auch den `mount`-Befehl verwenden wir analog zum Einhängen von `/dev/sdb1`.
Der Typ des Dateisystems wird automatisch erkannt.

```
hermann@deb-srv:~$ sudo mkdir /data2
hermann@deb-srv:~$ sudo mount /dev/sdb2 /data2
```

Prüfen, ob `/dev/sdb2` eingehängt ist

- mit `mount`

```
hermann@deb-srv:~$ mount | grep sdb2
/dev/sdb2 on /data2 type btrfs (rw,relatime,space_cache=v2,subvolid=5,subvol=/)
```

- mit `df`

```
hermann@deb-srv:~$ df -T | grep sdb2
/dev/sdb2      btrfs      5240832    5920   4699136     1% /data2
```

`/dev/sdb2` wieder aushängen

Auch `umount` verwenden wir analog zum Aushängen von `/dev/sdb1. Auf die Angabe des Dateisystemtyps verzichten wir.

- `umount` mit Gerätedatei

```
hermann@deb-srv:~$ mount | grep sdb2 # mounted? yes
/dev/sdb2 on /data2 type btrfs (rw,relatime,space_cache=v2,subvolid=5,subvol=/)
hermann@deb-srv:~$ sudo umount /dev/sdb2 # umount device
hermann@deb-srv:~$ mount | grep sdb2 # mounted? no
```

- `umount` mit Montierpunkt

```
hermann@deb-srv:~$ sudo mount /dev/sdb2 /data2 # mount again
hermann@deb-srv:~$ mount | grep sdb2 # mounted? yes
/dev/sdb2 on /data2 type btrfs (rw,relatime,space_cache=v2,subvolid=5,subvol=/)
hermann@deb-srv:~$ sudo umount /data2 # umount mountpoint
hermann@deb-srv:~$ mount | grep sdb2 # mounted? no
```

Automatisches Einhängen: `/etc/fstab`

Die Datei `/etc/fstab` enthält Informationen über alle Dateisysteme, die beim Systemstart automatisch eingehängt werden sollen.

Auch für die bei der Installation von Debian erstellten Partitionen `/dev/sda1`, `/dev/sda2` und `/dev/sda3` wurden Einträge in `/etc/fstab` erstellt.

Jeder Eintrag in `/etc/fstab` besteht aus sechs Feldern, die durch Leerzeichen oder Tabulatoren getrennt sind.

Zum Inhaltsverzeichnis ...

Felder in `/etc/fstab`

- **Dateisystem** - Gerätedatei oder UUID des Dateisystems
- **Montierpunkt** - Verzeichnis, in das das Dateisystem eingehängt wird
- **Typ** - Dateisystemtyp (`ext4`, `btrfs`, `vfat`, `nfs`, `cifs` etc.)
- **Optionen** - Optionen für das Einhängen des Dateisystems (z.B. `rw`, `ro` etc.). Die Optionen können auch spezielle Optionen für den Typ des einzuhängenden Dateisystems sein.
- **Dump** - Dump-Backup-Option (0 = kein Backup, 1 = Backup)
- **Pass** - Reihenfolge der Dateisystemüberprüfung beim Systemstart (0 = nicht überprüfen)

Beispiel-Einträge in `/etc/fstab` für `/var` und `/home`

```
# /home was on /dev/sda6 during installation
UUID=c59a3ba7-81d6-4177-8aa1-ff678e1f53c9 /home ext4 defaults 0 2
# /var was on /dev/sda3 during installation
UUID=20876574-a24f-43d4-a7e0-11a6a4e9417d /var  ext4 defaults 0 2
```

Diese Einträge wurden bei der Installation von Debian automatisch erstellt. Sie dienen uns als Vorlage für die Einträge für `/dev/sdb1` und `/dev/sdb2`.

Statt der UUID können auch die Gerätedateien verwendet werden, z.B. `/dev/sdb1` und `/dev/sdb2`.

Neue Einträge in `/etc/fstab` für `/dev/sdb1` und `/dev/sdb2`

```
# mount /dev/sdb1 to /data1
/dev/sdb1 /data1 ext4 defaults 0 2
# mount /dev/sdb2 to /data2
/dev/sdb2 /data2 btrfs defaults 0 2
```

Zum Ändern der `/etc/fstab`-Datei verwenden wir einen Texteditor wie `nano` oder `vim`. Die Datei ist nur für Benutzer mit Administratorrechten schreibbar. Deshalb verwenden wir `sudo`.

```
hermann@deb-srv:~$ sudo nano /etc/fstab
```

Einhängen mit `sudo mount -a`

Unsere neuen Dateisysteme sind eingetragen. Mit dem Kommando `sudo mount -a` können wir die Dateisysteme einhängen. `mount -a` liest die Datei `/etc/fstab` und hängt alle dort eingetragenen Dateisysteme ein, falls sie noch nicht eingehängt sind.

```
hermann@deb-srv:~$ sudo mount -a
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
```

Das klappt nicht auf Anhieb. Die Fehlermeldung gibt uns den Hinweis, dass wir mit `systemctl daemon-reload` den Systemd-Daemon neu laden müssen.

```
hermann@deb-srv:~$ sudo systemctl daemon-reload
hermann@deb-srv:~$ sudo mount -a
hermann@deb-srv:~$ mount | grep sdb
/dev/sdb1 on /data1 type ext4 (rw,relatime)
/dev/sdb2 on /data2 type btrfs (rw,relatime,space_cache=v2,subvolid=5,subvol=/)
hermann@deb-srv:~$ df -T | grep sdb
/dev/sdb1      ext4       5074592      24   4796040    1% /data1
/dev/sdb2      btrfs      5240832    5920   4699136    1% /data2
```

Die neuen Dateisysteme `/dev/sdb1` und `/dev/sdb2` sind nun eingehängt.

Automatisches Einhängen beim Systemstart testen

Um sicher zu gehen, dass diese Mounts auch beim Systemstart automatisch erfolgen, starten wir unser Debian-System neu.

```
hermann@deb-srv:~$ sudo reboot now
```

Nach dem Neustart prüfen wir noch einmal, ob die Dateisysteme `/dev/sdb1` und `/dev/sdb2` automatisch eingehängt wurden.

```
hermann@deb-srv:~$ df -T | grep sdb
/dev/sdb1      ext4         5074592      24   4796040    1% /data1
/dev/sdb2      btrfs        5240832    5920   4699136    1% /data2
```

Einhängen mit UUID statt Gerätedatei

In `/etc/fstab` können wir auch die UUID des Dateisystems verwenden, um es einzuhängen. Die UUID ist eindeutig und ändert sich nicht, selbst wenn die Gerätedatei sich ändert. Die UUID wurde bei der Erstellung des Dateisystems auf der Partition erstellt und ist in den Metadaten des Dateisystems gespeichert.

(Die Gerätedateien können sich ändern, wenn z.B. ein USB-Stick eingesteckt wird oder die Reihenfolge der Festplatten im System geändert wird, oder wenn auf einer Festplatte Partitionen hinzugefügt oder gelöscht werden. Die UUID bleibt jedoch gleich.)

UUID eines Dateisystems anzeigen mit `blkid`

- Ohne Argumente zeigt `blkid` alle lokalen Blockgeräte und deren UUIDs an.

```
hermann@deb-srv:~$ sudo blkid
/dev/sdb2: UUID="2e1968f6-fa95-4c89-aae9-27089aefc4da" ... TYPE="btrfs" ...
/dev/sdb1: UUID="1d396ff6-cda5-4f78-8324-1a118cbdb574" ... TYPE="ext4" ...
/dev/sda4: UUID="1acbce87-5460-4059-b188-58fca94eb075" TYPE="swap" ...
/dev/sda2: UUID="66e54599-4f05-4d53-a129-871f185c3597" ... TYPE="ext4" ...
/dev/sda5: UUID="01efcbd6-d8c0-49c7-9943-f42722d2009f" ... TYPE="ext4" ...
/dev/sda3: UUID="20876574-a24f-43d4-a7e0-11a6a4e9417d" ... TYPE="ext4" ...
/dev/sda1: UUID="FA07-070E" BLOCK_SIZE="512" TYPE="vfat" ...
/dev/sda6: UUID="c59a3ba7-81d6-4177-8aa1-ff678e1f53c9" ... TYPE="ext4" ...
```

- Mit Argumenten zeigt `blkid` nur die UUIDs der angegebenen Gerätedateien an.

```
hermann@deb-srv:~$ sudo blkid /dev/sdb1 /dev/sdb2
/dev/sdb1: UUID="1d396ff6-cda5-4f78-8324-1a118cbdb574" ... TYPE="ext4" ...
/dev/sdb2: UUID="2e1968f6-fa95-4c89-aae9-27089aefc4da" ... TYPE="btrfs" ...
```

Wir haben die UUIDs der Dateisysteme `/dev/sdb1` und `/dev/sdb2` ermittelt. Per Copy&Paste können wir die UUIDs statt der Gerätedateien in `/etc/fstab` eintragen.

Einträge in `/etc/fstab` mit UUIDs

```
# mount /dev/sdb1 to /data1
UUID=1d396ff6-cda5-4f78-8324-1a118cbdb574 /data1 ext4 defaults 0 2
# mount /dev/sdb2 to /data2
UUID=2e1968f6-fa95-4c89-aae9-27089aefc4da /data2 btrfs defaults 0 2
```

Nun können wir die Montierpunkte `/data1` und `/data2` aushängen. Beim Wiedereinhängen mit `sudo mount -a` werden die geänderten Einträge mit den UUIDs verwendet.

Immer wenn wir Änderungen an `/etc/fstab` vorgenommen haben, müssen wir vor dem Einhängen den Systemd-Daemon neu laden.

```
ermann@deb-srv:~$ df -T | grep data # /data1 and /data2 still mounted
/dev/sdb1      ext4        5074592      24   4796040    1% /data1
/dev/sdb2      btrfs       5240832    5920   4699136    1% /data2
hermann@deb-srv:~$ sudo umount /data1 # unmount /data1
hermann@deb-srv:~$ sudo umount /data2 # unmount /data2
hermann@deb-srv:~$ df -T | grep data # /data1 and /data2 unmounted
hermann@deb-srv:~$ sudo mount -a # systemd not reloaded since last change
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
hermann@deb-srv:~$ sudo systemctl daemon-reload # reload systemd
hermann@deb-srv:~$ sudo mount -a # mount /data1 and /data2 with UUIDs
hermann@deb-srv:~$ df -T | grep data # /data1 and /data2 mounted again
/dev/sdb1      ext4        5074592      24   4796040    1% /data1
/dev/sdb2      btrfs       5240832    5920   4699136    1% /data2
```

Automatisches Einhängen beim Systemstart testen

Wieder testen wir nach der Umstellung auf UUIDs, ob die Dateisysteme `/dev/sdb1` und `/dev/sdb2` beim Systemstart automatisch eingehängt werden.

```
hermann@deb-srv:~$ sudo reboot now
```

Nach dem Neustart prüfen ...

```
hermann@deb-srv:~$ df -T | grep data
/dev/sdb1      ext4       5074592      24   4796040    1% /data1
/dev/sdb2      btrfs      5240832    5920   4699136    1% /data2
```

Verwendung der neuen Dateisysteme

Die neuen Dateisysteme `/dev/sdb1` und `/dev/sdb2` sind jetzt einsatzbereit. Wir können sie wie jedes andere Dateisystem verwenden.

Verwendbar sind sie allerdings nur für den Benutzer `root`. Nur er hat die Schreibrechte auf `/data1` und `/data2`.

root könnte jetzt bei Bedarf Unterverzeichnisse für andere Benutzer anlegen und dann die Eigentümerschaft an den Verzeichnissen an diese Benutzer übertragen mit dem Kommando `chown`.