

Der Unix Werkzeugkasten

file

Mit dem Befehl `file` kann der Typ einer Datei ermittelt werden:

```
lothar@ubuntu:/tmp$ file readme.txt
```

```
readme.txt: ASCII text
```

file

Mit Hilfe des Befehls kann auch festgestellt werden, für welche Plattform eine ausführbare Datei übersetzt wurde:

```
lothar@ubuntu:/tmp$ file /bin/ls
```

```
/bin/ls: ELF 32-bit LSB executable, Intel 80386,  
version 1 (SYSV), dynamically linked (uses  
shared libs), for GNU/Linux 2.6.24,  
BuildID[sha1]=0x274c7a324a48f28c5c5f80cfbbd9bece  
c6bcebe5, stripped
```

Die Datei **ls** hat also das Executable and Linked Format (ELF) und wurde für 32-Bit Intel-Prozessoren übersetzt.

cat

Mit dem Befehl **cat** wird der Inhalt einer Datei auf der Standardausgabe ausgegeben:

```
lothar@ubuntu:/tmp$ cat readme.txt
```

```
readme line 1  
readme line 2  
readme line 3
```

Bei großen Dateien wird die Ausgabe schnell unübersichtlich. Oft will man nur wissen, ob ein bestimmter Text in einer Datei enthalten ist.

grep

Mit dem Befehl **grep** wird der Inhalt einer Datei nach einem Textmuster durchsucht.

Alle Zeilen aus der Datei **readme.txt** anzeigen, die das Zeichen **2** enthalten:

```
lothar@ubuntu:/tmp$ grep 2 readme.txt
```

```
readme line 2
```

Es werden alle Zeilen angezeigt, die einen Treffer liefern.

grep ist sehr mächtig.

Als Suchtexte und für Dateinamen können auch reguläre Ausdrücke verwendet werden.

grep

Mit Hilfe der Option **-l** werden nicht die Trefferzeilen, sondern die Dateinamen angezeigt:

```
lothar@ubuntu:/tmp$ grep -l 2 *.txt
```

```
readme.txt
```

Es werden alle Dateien mit der Erweiterung **txt** nach dem Text **2** durchsucht.

more und tail

`cat` gibt die ganze Datei aus. Das Befehl `more` leistet das Gleiche, pausiert aber nach jeweils einer Bildschirmseite. Die Ausgabe wird mit durch Tastendruck fortgesetzt.

Oft (etwa bei Logdateien) interessieren uns nur die letzten Zeilen einer Datei:

```
lothar@ubuntu:/tmp$ tail /var/log/syslog -n 20
```

Pipes

Seine volle Mächtigkeit entfaltet Unix durch die Verknüpfung von Befehlen mit Hilfe von Pipes:

Die Ausgabe eines Befehls wird vom zweiten Befehl verarbeitet. Für die Verknüpfung wird das Pipezeichen `|` verwendet:

```
lothar@ubuntu:/tmp$ ls /bin | grep mount
```

Findet alle Dateien im Verzeichnis `/bin`, deren Namen den Text `mount` enthält:

```
fusermount  
mount  
mountpoint  
umount
```


find

Oft weiss man nicht, in welchem Verzeichnis eine Datei liegt.
Hier hilft **find**:

```
lothar@ubuntu:/tmp$ find . -name readme.txt
```

Das aktuelle Verzeichnis und alle Unterverzeichnisse wird
nach der Datei **readme.txt** durchsucht.

sudo

Interessant ist, dass das Beispiel zu `find` hier einen Fehler liefert:

```
./readme.txt  
find: `./pulse': Permission denied
```

Wir finden den Grund mit Hilfe von `ls -l`:

```
drwx----- 2 root    root    4096 Mar  3 08:16 pulse
```

Nur der Superuser hat das Recht in das Verzeichnis `pulse` zu wechseln.

sudo

Einige Benutzer dürfen Befehle als Superuser ausführen:

```
lothar@ubuntu:/tmp$ sudo find . -name readme.txt
```

Nachdem Eingabe des Passwortes des Benutzers, wird der Befehl fehlerfrei ausgeführt.

Warnung:

Melden Sie sich so selten wie möglich explizit als Superuser an! Die Wahrscheinlichkeit, dass Sie irreparablen Schaden an Ihrem System anrichten wächst von Mal zu Mal! Nutzen Sie besser **sudo**!

find und grep zusammen

Zum Befehl **find** gibt es die Option **-exec**. Die gefundenen Dateien können mit einem weiteren Befehl verarbeitet werden:

```
find . -type f -exec grep -l line {} \;
```

Findet alle Dateien (**-type f**[ile]), die den Text **line** enthalten.

Die geschweiften Klammern sind Platzhalter für die gefundenen Dateien.

Mit dem Semikolon wird der zu **-exec** gehörende Befehl abgeschlossen. Das Semikolon muss per **** maskiert werden.

strings

Gelegentlich will man wissen, welche (lesbaren) Texte eine **Binärdatei** enthält. Editoren sind hier wenig hilfreich. Dafür gibt es den Befehl **strings**:

```
lothar@ubuntu:~$ strings /bin/ls | grep error
```

Findet alle Texte in der Datei **/bin/ls**, die das Wort **error** enthalten:

```
error
error initializing
month strings
write error
```

which

Es kann vorkommen, dass es mehrere ausführbare Dateien mit dem gleichen Namen gibt. Es kann dann Unsicherheit bestehen, welche Datei überhaupt ausgeführt wird. Hier hilft **which**:

```
lothar@ubuntu:~$ which ls
```

```
/bin/ls
```

ps

Eine Liste laufenden Prozesse erhält man mit dem Befehl **ps** (processing status).

Ohne Optionen werden nur die Prozesse gezeigt, die der Nutzer gestartet hat.

Mit der Option **aux** werden alle Prozesse angezeigt:

```
lothar@ubuntu:~$ ps aux | grep bash
```

lothar	1441	0.0	0.9	7356	3768	tty1	S+	16:37	0:00	-bash
lothar	1787	0.0	0.8	7332	3536	pts/0	Ss+	16:39	0:00	-bash
lothar	1982	0.0	0.9	7404	3912	pts/2	Ss	16:41	0:00	-bash

In den meisten Fällen wird das Ergebnis, wie hier, über eine Pipe weiter verarbeitet.

top

Hilfreich ist auch der Befehl `top`. Es werden nicht nur die Prozesse die die meisten Ressourcen verbrauchen, sondern auch allgemeine Systeminformationen, wie genutzter Hauptspeicher oder Anzahl der Prozesse angezeigt:

```
top - 18:02:16 up 1:25, 5 users, load average: 0.00, 0.01, 0.05
Tasks: 107 total, 1 running, 106 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 394280 total, 229380 used, 164900 free, 61896 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free, 74292 cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2691	lothar	20	0	5204	1336	1012	R	0.3	0.3	0:00.20	top
1	root	20	0	3576	2040	1284	S	0.0	0.5	0:01.44	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

Everything is a File

„Everything is a File“ beschreibt eine der definierenden Eigenschaften von Unix.

Auch Geräte und Speichermedien werden durch Dateien repräsentiert. Diese findet man im Verzeichnis `/dev`

```
lothar@ubuntu:~$ ls /dev | wc
```

```
216      216    1316
```

`wc` (word count) findet 216 Einträge im Verzeichnis. Jeder Eintrag entspricht einem Gerätetreiber (im weitesten Sinne). Ein physikalisches Gerät ist nicht notwendigerweise verfügbar.

Speichermedien

Will man wissen, welche Speichermedien es in einem System gibt, ist es schwierig sich im `/dev`-Verzeichnis zu orientieren. Hier hilft `fdisk`. Der Befehl ist eigentlich zum verwalten von Partitionen eines Speichermediums vorgesehen, hilft uns aber hier mit der Option `ls` weiter:

```
lothar@ubuntu:~$ sudo fdisk -ls
```

```
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000cd608
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	39845887	19921920	83	Linux
/dev/sda2		39847934	41940991	1046529	5	Extended
/dev/sda5		39847936	41940991	1046528	82	Linux swap /

Speichermedien

Unser System erkennt ein Speichermedium (`/dev/sda`) mit den drei Partitionen `sda1`, `sda2` und `sda5`.

Für jede Partition wird der Typ des Dateisystems (Linux, Extended und Swap) mit angezeigt.

Speichermedien

Wenn wir eine neue SD-Karte einlegen, liefert **fdisk** zusätzlich die folgende Information:

```
Disk /dev/sdb: 4014 MB, 4014997504 bytes
1 heads, 32 sectors/track, 245056 cylinders, total 7841792 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xa7302271
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		2048	7841791	3919872	b	W95 FAT32

Wir können über **/dev/sdb** auf die Karte zugreifen. Die Karte wurde bereits formatiert und enthält eine Partition mit einem FAT32-Dateisystem.

Speichermedien

Einen guten Überblick über verfügbare Speichermedien (block devices) liefert auch der Befehl `lsblk`:

```
hfu@host00:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   20G  0 disk
├─sda1       8:1    0   19G  0 part /
├─sda2       8:2    0    1K  0 part
└─sda5       8:5    0 1022M  0 part [SWAP]
sde          8:64    1 14.5G  1 disk
├─sde1       8:65    1    56M  1 part
└─sde2       8:66    1 14.4G  1 part
sr0         11:0    1 1024M  0 rom
```

Speichermedien

Warnung:

Aus der Ausgabe von `fdisk -ls` oder `lsblk` geht nicht klar hervor, welches Speichermedium zu welchem Eintrag in `/dev` gehört.

Auf der sicheren Seite ist man, wenn man den Befehl unmittelbar **vor und nach** dem einlegen ausführt und die Ausgaben vergleicht.

Profi-Tipp

Der Linux-Kernel überwacht das einlegen und abziehen von Geräten. Nachdem man eine SD-Karte eingelegt hat liefert der Befehl **dmesg** (driver message) die folgende Ausgabe:

```
[ 1637.585010] mmc0: card b368 removed  
[ 1711.835423] mmc0: new high speed SDHC card at address b368  
[ 1711.835801] mmcblk0: mmc0:b368 NCard 3.73 GiB  
[ 1711.837143] mmcblk0: p1 p2
```

Die eingelegte Karte wird als **/dev/mmcblk0** geführt und hat die beiden Partitionen **/dev/mmcblk0p1** und **/dev/mmcblk0p2**

Speichermedien

Selbst wenn wir auf unserer SD-Karte auf der Partition `/dev/sdb1` ein Dateisystem haben, so kann dies noch nicht genutzt werden. Befehle wie der folgende schlagen fehl:

```
lothar@ubuntu:~$ sudo mkdir /dev/sdb1/newdir
```

```
mkdir: cannot create directory '/dev/sdb1/newdir': Not a directory
```

Die Einträge im `/dev`-Verzeichnis können nicht wie die Verzeichnisse eines normalen Dateisystems genutzt werden!

mount

Um das Dateisystem einer formatierten Partitionen nutzen zu können, müssen sie in ein existierendes Verzeichnis **eingehängt** (gemountet) werden. Der Befehl **mount** liefert eine Liste aller eingehängten Dateisysteme. In unserem System erhält diese Liste auch den folgenden Eintrag:

```
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
```

Das Dateisystem auf **/dev/sda1** wurde also unter dem Verzeichnis **/** eingehängt. Es ist das so genannte Root-Dateisystem.

mount

Wir hängen unsere SD-Karte ein. Dazu muss zunächst ein Verzeichnis erstellt werden. Per Konvention liegen diese sogenannten Mount-Punkte im Verzeichnis `/mnt`

```
lothar@ubuntu:~$ sudo mkdir /mnt/sdcard  
lothar@ubuntu:~$ mount /dev/sdb1 /mnt/sdcard
```

Wir prüfen, ob der Befehl erfolgreich ausgeführt wurde:

```
lothar@ubuntu:~$ mount
```

```
/dev/sdb1 on /mnt/sdcard type vfat (rw)
```

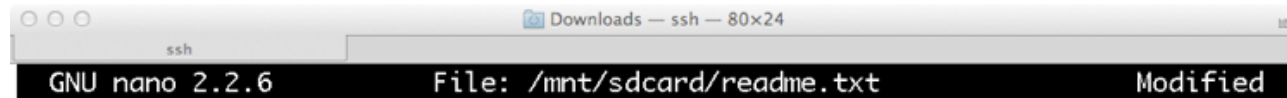
Das Dateisystem wurde eingehängt, der Typ `vfat` wurde automatisch erkannt.

Der Editor nano

Wir legen eine Datei auf unserer (eingehängten) SD-Karte an, indem wir den intuitiv bedienbaren Editor **nano** nutzen:

```
lothar@ubuntu:~$ sudo nano /mnt/sdcard/readme.txt
```

nano



```
GNU nano 2.2.6      File: /mnt/sdcard/readme.txt      Modified
```

This a new file in my SD card

```
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Die Befehle im unteren Bereich werden über die CTRL-Taste aktiviert. Mit Exit kann die Datei gesichert werden.

umount

Eingehängte Dateisysteme können auch wieder **ausgehängt** werden:

```
lothar@ubuntu:~$ sudo umount /mnt/sdcard/
```

Prüfen Sie anschließend mit **mount**, ob die Operation erfolgreich war.

Tipps:

1. Hängen Sie Dateisysteme immer aus, bevor Sie die Speichermedien entfernen. Sie können Ihr Dateisystem sonst irreparabel beschädigen.
2. Achten Sie darauf, dass Sie sich beim aushängen nicht im eingehängten Verzeichnis befinden. Sie erhalten ansonsten die Fehlermeldung **device is busy**.

Images

Von Datenträgern können auch binäre Abbilder (Images) angefertigt werden. Es handelt sich dabei nicht um Kopien von *Dateien*, sondern um binäre Kopien von Partitionen.

Unsere *ausgehängte* SD-Karte kopieren wie folgt in eine Datei:

```
lothar@ubuntu:~/tmp$ sudo cp /dev/sdb sdcard.img
```

Das Dateisystem auf der Karte enthält nur eine kleine Datei. Es dauert aber einige Minuten, bis das Image angefertigt wurde, da der ganze Karteninhalt (z.B. 4GB) kopiert wird.

Images

Wir schauen uns das Image genauer an. Der Befehl `ls -l` liefert uns die Größe der Image-Datei

```
-rw-r----- 1 root root 4014997504 Mar  3 19:17 sdcard.img
```

Der Befehl `file` erkennt, dass es sich um eine Image-Datei handelt und liefert weitere Details:

```
sdcard.img: x86 boot sector; partition 1: ID=0xb, starthead  
33, startsector 2048, 7839744 sectors, extended partition  
table (last)\011, code offset 0x0
```