## 1.3 Anwendung/Dateisystem

### 1.3.1 Verzeichnisse und Dateien

Nach dem ersten Kennenlernen der wichtigsten Befehle soll es nun darum gehen, mit Hilfe der Befehle `cd` und `ls` weitere Kenntnisse über die Verzeichnisstruktur zu bekommen.

Zunächst gibt es eine Besonderheit bim Befehl `ls`: Wie bereits oben beschrieben, kann man als Parameter nicht nur Dateinamen, sondern auch Verzeichnisnamen in beliebiger Reihenfolge als Parameter angeben. Allerdings ist die Anzeige dann aber verschieden:

- Bei einem Dateinamen als Parameter wird der *Name* der Datei angezeigt; bei Angabe der Option `-l` erhält man zusätzlich die Attribute (=Eigenschaften)) der Datei (dazu später mehr)
- Bei einem Verzeichnisnamen als Parameter wird der *Inhalt* des Verzeichnisses angezeigt, also die Namen der enthaltenen Dateien und Unterverzeichnisse; bei Angabe der Option `-l` erhält man zusätzlich die Attribute der enthaltenen Dateien und Unterverzeichnisse, außerdem den Speicherplatz in kiB-Blöcken
- Gibt man keinen Parameter an, wird der Inhalt des aktuellen Verzeichnisses angezeigt

Hier ein Beispiel:

Terminal

```
schueler@debian964:~$ touch a.html           # Datei anlegen
schueler@debian964:~$ ls -l a.html           # Name+Attribute anzeigen
-rw-r--r-- 1 schueler fet4u  0  1. Okt 16:03 a.html
schueler@debian964:~$ mkdir meinverz         # Verzeichnis erstellen
schueler@debian964:~$ touch meinverz/b.html # Datei darin anlegen
schueler@debian964:~$ ls -l meinverz/        # Verz.-Inhalt anzeigen
insgesamt 0
-rw-r--r-- 1 schueler fet4u  0  1. Okt 16:06 b.html
```

Mit der Option `-d` kann man bei Verzeichnissen bewirken, dass nicht der Inhalt, sondern das Verzeichnis selbst angezeigt wird:

Terminal

```
schueler@debian964:~$ ls -l -d meinverz/   # Name+Attribute anzeigen
drwxr-xr-x 2 schueler schule 4096  1. Okt 16:06 meinverz/
```

| | Datei `dat` | Verzeichnis `verz` |
|---|---|---|
| Name anzeigen | `ls dat` | `ls -d verz` |
| Name und Attribute | `ls -l dat` | `ls -l -d verz` |
| Inhalt anzeigen | `cat dat` | `ls -l verz` |

Tabelle 1: Anzeigen von Name, Eigenschaften und Inhalt bei Dateien und Verzeichnissen

Bei einer Datei als Parameter bewirkt die Option `-d` nichts, aber sie schadet auch nicht. Die Tabelle1 fasst das Ergebnis noch einmal zusammen.

### 1.3.2 Optionen – ein Nachtrag

Bei `ls` und den meisten anderen Befehlen darf man alle die Optionen, die außer dem Minuszeichen nur einen Buchstaben haben, zusammenfassen. Die beiden folgenden Befehlszeilen sind deshalb in ihrer Bedeutung gleich:

Terminal
```
schueler@debian964:~$ ls -l -d meinverz/
drwxr-xr-x 2 schueler schule 4096  1. Okt 16:06 meinverz/
schueler@debian964:~$ ls -ld meinverz/
drwxr-xr-x 2 schueler schule 4096  1. Okt 16:06 meinverz/
```

Manche Befehle haben auch Optionen, die aus mehr als einem Buchstaben bestehen, die sogenannten Lang-Optionen. Lang-Optionen kann man nicht zusammenfassen. Meistens (leider nicht immer) werden die Lang-Optionen durch zwei Minuszeichen eingeleitet:

Terminal
```
schueler@debian964:~$ ls -l --directory meinverz/
drwxr-xr-x 2 schueler schule 4096  1. Okt 16:06 meinverz/
```

Hier bedeutet `--directory` dasselbe wie `-d`. Einige Befehle haben nur lange, andere nur kurze Optionen und bei vielen Befehlen gibt es zu jeder langen eine gleichbedeutende kurze Option[1].

### 1.3.3 Dateiattribute

Nach der Eingabe von `ls -l dateiname` (für eine Datei) oder `ls -ld verzeichnisname` (für ein Verzeichnis) erhält man eine Zeile mit den sogenannten *Attributen* (=Eigenschaften) dieser Datei bzw. dieses Verzeichnisses:

Terminal
```
schueler@debian964:~$ ls -l buch.txt
-rw-r--r-- 1 schueler fet4u 1285 16. Sep 08:44 buch.txt
schueler@debian964:~$ ls -ld geda
drwxr-xr-x 3 schueler fet4u  208 16. Sep 19:43 geda
```

In Tabelle 2 sind diese Attribute einmal erläutert. In Spalte 1 der Ausgabe von `ls -l` geht es

| Spalte | Beispiel | Bedeutung | Bedeutung hier |
|---|---|---|---|
| 1 | `-` | Art des Dateisystemobjekts | Es ist eine Datei |
| 2 | `rw-r-r-` | Berechtigungen | Wird hier nicht beschrieben |
| 3 | `1` | Link-Count | Datei hat 1 Namen |
| 4 | `schueler` | Besitzer | `schueler` ist Besitzer |
| 5 | `fet4u` | Besitz-Gruppe | `fet4u` ist Gruppe |
| 6 | `1285` | Größe in Bytes | 1285 Bytes |
| 7 | `16. Sep` | mtime-Datum | Sie wurde am 16.09.2024 modifiz., |
| 8 | `08:44` | mtime-Zeit | und zwar um 8.44 Uhr |
| 9 | `buch.txt` | Name des Dateisystemobjekts | Die Datei heißt `buch.txt` |

Tabelle 2: Attribute einer Datei oder eines Verzeichnisse, die `ls -l` anzeigt

meistens darum, ob es sich um eine Datei oder ein Verzeichnis handelt. Es gibt aber noch weitere Arten von Dingen im Verzeichnisbaum. Sie sind in Tabelle 3 zu sehen. Dateien, Verzeichnisse und diese anderen Dinge fasst man übrigens unter dem Namen *Dateisystemobjekt* zusammen[2]. Jenseits von `ls -l` werden Dateien meistens nicht mit dem Minuszeichen, sondern mit dem Buchstaben `f` (=*file*) gekennzeichnet.

In den Spalten 2, 4 und 5 geht es um die Berechtigungen für dieses Dateisystemobjekt, also darum, wer was mit diesem Objekt machen darf.

Die Spalte 6 gibt die Größe einer Datei in Bytes an; bei einem Verzeichnis wird die Größe des Verzeichnisses selbst (nicht die der enthaltenen Objekte) auf eine volle Blockgröße aufgerundet angegeben.

[1] Es gibt auch Optionen, die selbst einen Parameter haben, der entweder durch Leerzeichen oder durch ein Gleichheitszeichen von der Option getrennt ist. Das kann man dann in der Manual-Page nachlesen.

[2] In Manual-Pages oft irreführend als „Dateityp“ bezeichnet

| Anzeige (`ls -l`) | Bedeutung | Englisch | Beispiel |
|---|---|---|---|
| `-` | Datei | *regular file* | `/etc/passwd` |
| `d` | Verzeichnis | *directory* | `/etc` |
| `b` | blockorientiertes Gerät | *block device* | `/dev/sda1` (HDD-Partition) |
| `c` | zeichenorientiertes Gerät | *char device* | `/dev/ttyS0` (serielle Schnittst.) |
| `l` | symbolischer Link | *link* | `/usr/bin/cc` (C-Compiler) |
| `p` | FIFO | *named pipe* | `/dev/initctl` |
| `s` | Socket | *socket* | `/var/run/cups/cups.sock` |

Tabelle 3: Arten von Dateisystemobjekten

Die Spalten 7 und 8 zeigen den Zeitpunkt der letzten Änderung an, den sogenannten Zeitstempel[3].

Und die letzte Spalte zeigt den relativen Pfadnamen des Dateisystemobjekts[4].

### 1.3.4 Hard-Links

In Spalte 3 wird der Link-Count mitgeteilt. Das ist die Anzahl der sogenannten *Hard-Links* auf eine Datei. Hard-Links sind einfach verschiedene gleichberechtigte Namen derselben Datei (Abbildung 1, linke Seite). Die Erzeugung eines neuen Namens geschieht mit dem `ln`-Befehl. Ändert man

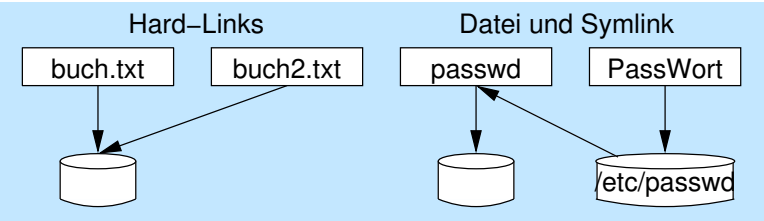


Abbildung 1: Hard-Links (linke Seite) und Symlinks (rechte Seite)

anschließend die Datei, werden die Änderungen unter allen Hard-Links gleichermaßen sichtbar:

Terminal

```
schueler@debian964:~$ ls -l buch.txt
-rw-r--r--  1 schueler fet4u 1285 16. Sep 08:44 buch.txt
schueler@debian964:~$ ln buch.txt buch2.txt
schueler@debian964:~$ ls -l buch2.txt
-rw-r--r--  2 schueler fet4u 1285 16. Sep 08:44 buch2.txt
schueler@debian964:~$ echo "Hallo" >> buch2.txt
schueler@debian964:~$ ls -l buch.txt buch2.txt
-rw-r--r--  2 schueler fet4u 1291 16. Sep 10:59 buch2.txt
-rw-r--r--  2 schueler fet4u 1291 16. Sep 10:59 buch.txt
schueler@debian964:~$
```

Löscht man einen der beiden Einträge, bleibt der Dateiinhalt bestehen. Löscht man aber den letzten Eintrag einer Datei, bekommt der Link-Count den Wert null. Das Betriebssystem weiß nun, dass niemand mehr diese Datei braucht. Daher löscht es den I-Node und alle betreffenden Datenblöcke[5].

[3] Es gibt mittlerweile drei weitere Zeitstempel, die aber normalerweise nicht angezeigt werden

[4] bei einem Verzeichnisinhalt relativ zum als Parameter angegeben Verzeichnis; bei den Objekten, die als Parameter angegeben wurden, relativ zum aktuellen Verzeichnis

[5] Sie werden nicht wirklich gelöscht, sondern auf der Partition als frei gekennzeichnet.

### 1.3.5 Symlinks

Hard-Links haben zwei Nachteile:

a) Hard-Links kann man nur für reguläre Dateien erstellen, nicht aber für Verzeichnisse. Diese Einschränkung wurde absichtlich gesetzt, damit der Verzeichnisbaum keine Schleifen (a ist Unterverzeichnis von b und b ist Unterverzeichnis von a) bekommt

b) Hard-Links sind nur innerhalb der gleichen Partition möglich, denn die Inode-Nummern werden für jede Partition getrennt gezählt

Um diese Einschränkungen zu umgehen, hat man eine zweite Art von Links, die sogenannten *symbolischen Links* (abgekürzt Symlinks, anderer Name: *Soft-Links*) ermöglicht (Abbildung 1, rechte Seite). Erzeugt werden sie mit der `-s`-Option des `ln`-Befehls.

Terminal

```
schueler@debian964:~$ ln -s /etc/passwd PassWort
schueler@debian964:~$ ls -l /etc/passwd PassWort
-rw-r--r-- 1 root     root  1306 14. Sep 20:15 /etc/passwd
lrwxrwxrwx 1 schueler fet4u   11 16. Sep 11:06 PassWort -> /etc/passwd
schueler@debian964:~$
```

Hier wird also ein neues Dateisystemobjekt erzeugt, das nur den Pfadnamen der Originaldatei enthält; es handelt sich quasi um einen Zeiger auf den Original-Pfadnamen. Dementsprechend darf der Pfadname absolut oder relativ angegeben werden, dabei sind die gleichen Konsequenzen wie bei HTML-Verweisen zu beachten. Meistens ist der absolute Pfadname zu empfehlen.

Wenn man wissen will, wohin ein Link zeigt, kann man (außer `ls`) die Befehle `readlink` und `realpath` verwenden:

Terminal

```
schueler@debian964:~$ touch a   # a ist eine Datei
schueler@debian964:~$ ln -s a b # b zeigt auf a
schueler@debian964:~$ ln -s b c # c zeigt auf b
schueler@debian964:~$ ls -l ?
-rw-r--r-- 1 schueler schueler 0 16. Sep 20:27 a
lrwxrwxrwx 1 schueler schueler 1 16. Sep 20:26 b -> a
lrwxrwxrwx 1 schueler schueler 1 16. Sep 20:27 c -> b
schueler@debian964:~$ readlink c
b
schueler@debian964:~$ realpath c
/home/schueler/a
```

### 1.3.6 Realisierung

Wie werden nun Verzeichnisse und Dateien auf einer Partition eines Massenspeichers abgelegt?

Zunächst zu den Verzeichnissen: Ein Verzeichnis kann man sich vorstellen (fast) wie eine Textdatei, in der die Namen der Dateien und Unterverzeichnisse gespeichert werden. Zusätzlich zum Namen enthält jeder Eintrag in einem Verzeichnis auch die Nummer des dazugehörigen I-Nodes (Abbildung 2). Mehr Information ist im Verzeichniseintrag nicht vorhanden (Ausnahme aus Performance-Gründen: siehe `man readdir`).

Die Trennung von Dateiname und Inhalt wird so vollzogen, dass der Ort des Datei-Inhalts und die Datei-Attribute (=Eigenschaften) getrennt vom Namen gespeichert werden, und zwar in einem sogenannten I-Node (=*information node* =Informations-Knotenpunkt). Abbildung 3 zeigt schematisch den Aufbau eines I-Nodes (mehr Informationen unter `man 3type stat` und `man inode`).

Die I-Node-Nummer eines Dateisystemobjekts kann man sich mit der `-i`-Option des `ls`-Befehls anzeigen lassen; dabei erkennt man wieder die Unterschiede zwischen Hard-Link (gleicher I-Node) und Symlink (extra I-Node für den Symlink):

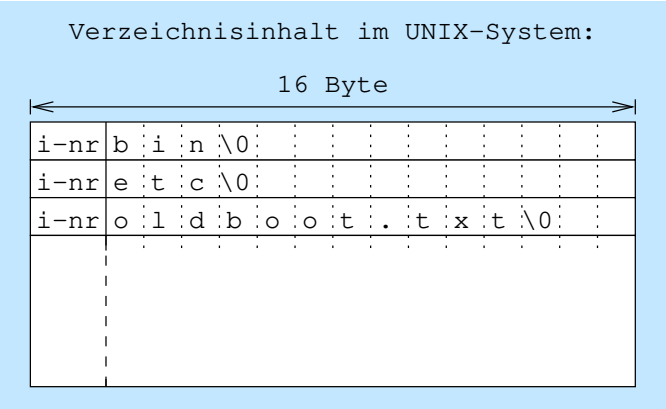


Abbildung 2: Aufbau eines Verzeichniseintrags

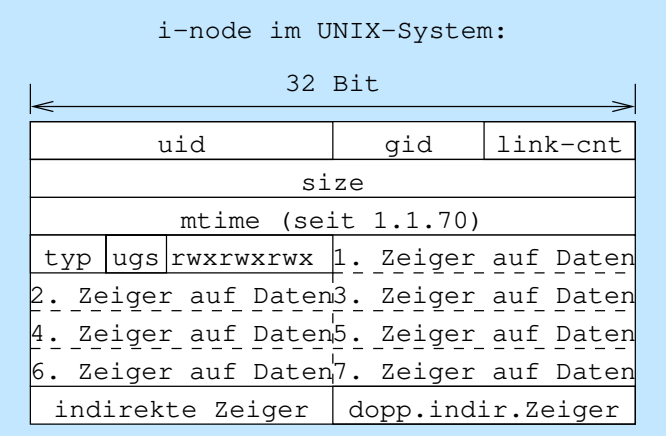


Abbildung 3: Aufbau eines I-Node

Terminal

```
schueler@debian964:~$ ls -li buch.txt buch2.txt /etc/passwd PassWort
 2420 -rw-r--r-- 2 schueler fet4u 1291 16. Sep 10:59 buch2.txt
 2420 -rw-r--r-- 2 schueler fet4u 1291 16. Sep 10:59 buch.txt
18543 -rw-r--r-- 1 root     root  1306 14. Sep 20:15 /etc/passwd
 7165 lrwxrwxrwx 1 schueler fet4u   11 16. Sep 11:06 PassWort -> /etc/passwd
```

Die Struktur der Partition ist also getrennt in:

a) die Datei- und Verzeichniseigenschaften, gespeichert in den I-Nodes

b) die Datei- und Verzeichnisinhalte, gespeichert in den Datenblöcken der Partition

Abbildung 4 zeigt nun, wie eine Partition aussehen kann: Ganz links ist meistens ein Bootblock. Daneben sieht man den Superblock, der Eigenschaften des Verzeichnisbaums enthält Es folgen eine Tabelle (*bitmap*), in der vermerkt ist, welche I-Nodes noch frei sind und eine zweite Tabelle, in der vermerkt ist, welche Datenblöcke frei sind. Nun kommt ein kleineres Bereich mit I-Nodes. Und ganz rechts liegen dann die Datenblöcke (also die Nutzlast der Partition).

Was passiert, wenn ein Anwendungsprogrammierer in seiner Programmiersprache die Datei `/etc/brief.txt` öffnet? Ohne, dass er es merkt, passiert das (Abbildung 4):

a) Im Superblock ist die Nummer des I-Nodes für das Stammverzeichnis `/`

b) Der I-Node des Stammverzeichnisses wird geöffnet

c) Im I-Node von `/` steht, wo die Daten des Stammverzeichisses liegen – sie enthalten die Verzeichniseinträge

d) Im Verzeichnisinhalt wird der Eintrag `etc` gesucht und gefunden – dort steht die I-Node-Nummer von `/etc`

e) Im I-Node von `/etc` steht, wo die Daten von `/etc` liegen – sie enthalten die Verzeichniseinträge

f) Im Verzeichnisinhalt von `/etc` wird der Eintrag `brief.txt` gesucht und gefunden – dort steht die I-Node-Nummer von `/etc/brief.txt`

g) Im I-Node von `/etc/brief.txt` steht, wo die Daten dieser Datei liegen – dort findet man den Dateiinhalt

Dies ist nur eine von mehreren möglichen Arten, Dateien und Verzeichnisse in einer Partition zu speichern. Festgelegt wurde diese Art bei der Formatierung der Partition. Danach kann man sie nicht mehr ändern, ohne die Daten zu löschen. Man sagt: Bei der Formatierung erhält die Partition ein bestimmtes *Dateisystem* [6]. Linux beherrscht viele verschiedene Dateisysteme [7]. Hier sind ein paar davon:

- `ext4` – unter Linux üblich
- `ext`, `ext2` und `ext3` – Vorläufer von `ext4`
- `minix` – sehr klein und übersichtlich, hier für die Abbildungen 3 und 2 verwendet
- `ISO 9660` – für optische Datenträger
- `FAT` – gerne für USB-Speichersticks verwendet
- `NTFS` – wie `FAT` aus der MS-Wind.-Welt

[6] Vorsicht: Leider wird der Begriff *Dateisystem* auch für den Verzeichnisbaum benutzt.

[7] Das ist eben der Vorteil eines offenen Systems: Keine Firmenpolitik, die einen davon abhält, Technik der Konkurrenz zu benutzen

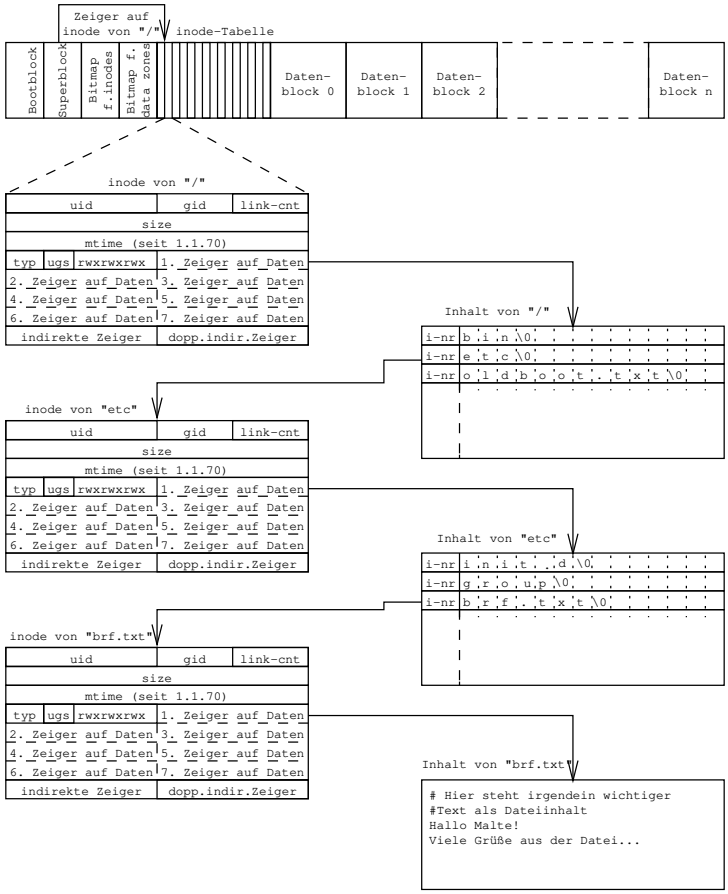


Abbildung 4: Öffnen der Datei `/etc/brief.txt`

Für die meisten Anwender ist es gleichgültig, welches Dateisystem auf einer Partition vorliegt. Die Unterschiede liegen hauptsächlich darin, welche Attribute unterstützt werden und welche Arten von Links möglich sind. So sind zum Beispiel Hard-Links beim FAT-Dateisystem technisch nicht möglich (weil FAT keine I-Nodes kennt).

Wenn es dagegen darum geht, möglichst viele Daten pro Zeit zu bewegen, kann die Wahl des richtigen Dateisystems und der richtigen Parameter beim Formatieren wichtig werden.