1.2 Anwendung/Befehle

1.2.1 Einloggen, Ausloggen und Umschalten der Bildschirme

Arbeitet man direkt an einem Linux-Computer, so trifft man nach dem Hochfahren eine von zwei Möglichkeiten an. Entweder man sieht einen GUI-Bildschirm mit Login-Möglichkeit oder einen Text-Bildschirm mit Login-Möglichkeit.

Im ersten Fall ist das Programm, das man sieht, der *Display Manager*[1]. Hier loggt man sich wie gewohnt mit Name und Passwort ein. Beim Eintippen des Passwortes wird für jedes Zeichen ein Punkt angezeigt[2]. Ausloggen kann man sich über den passenden Menüpunkt.

Im zweiten Fall sieht man die Oberfläche des Programms `login`. Auch hier loggt man sich mit Name und Passwort ein. Beim Eintippen des Passwortes wird aber nichts angezeigt.

Terminal

```
debian964 login: schueler
Passwort:
Letzte Anmeldung: Montag, den 27. Mai 2024, 23:05:49 CET auf tty1
...
schueler@debian964:~$
```

Ausloggen kann man sich, indem man Strg + D drückt oder durch Eintippen von `logout`[3], gefolgt von der Taste ↩ oder Enter .

Am Text-Bildschirm gibt es verschiedene Funktionstasten, die das Umschalten auf weitere Text-Bildschirm und gegebenenfalls einen oder mehrere GUI-Bildschirme gestatten (siehe Tabelle 1). An wie vielen und an welchen Funktionstasten jetzt tatsächlich Text- oder Graphik-Bildschirme

Ziel	Im Text-Bildschirm	Im Graphik-Bildschirm
Text-Bildschirm 1	Alt + F1	Strg + Alt + F1
Text-Bildschirm 2	Alt + F2	Strg + Alt + F2
Text-Bildschirm 3	Alt + F3	Strg + Alt + F3
...	...	...
Graphik-Bildschirm 2	Alt + F7	Strg + Alt + F7
Graphik-Bildschirm 1	Alt + F8	Strg + Alt + F8
Text-Bildschirm 9	Alt + F9	Strg + Alt + F9
...	...	...
Text-Bildschirm 12	Alt + F12	Strg + Alt + F12
Neustart Graphik-Bildschirm	—	Strg + Alt + ←
System Request	—	Alt - Druck + Taste

Tabelle 1: Tastenkombinationen zum Umschalten zwischen Bildschirmen

liegen, ist vom Administrator konfigurierbar. Mit Strg + Alt + ← kann man einen Graphik-Bildschirm schnell beenden; der System Request dient dazu, vom Graphik-Bildschirm aus über einzelne Funktionstasten das System schnell zu verwalten.

Eine Besonderheit ist noch bei PC-Tastaturen unter Linux zu beachten: Die Zeichen `@`, ~ und \ sind bei der deutschsprachigen Tastatur (wie bei Windows) mit AltGr zu bekommen, aber (anders als bei Windows) nicht mit Strg + Alt (siehe Tabelle 2).

[1] Der bei Debian übliche Display Manager `gdm` gibt einem die Möglichkeit, zwischen mehreren Benutzeroberflächen (*Window Manager* genannt) auszuwählen. Die folgenden Beispiele beziehen sich auf die Oberfläche *GNOME Classic*.

[2] Oder drei Punkte oder gar nichts. Das ist einstellbar

[3] oder `exit`

Zeichen	Tastenkombination
@	AltGr + Q
~	AltGr + +
\	AltGr + ß

Tabelle 2: Linux-Tastenkombinationen für wichtige Zeichen

1.2.2 Programme aufrufen mit der Textkonsole

Das Besondere an Linux ist, dass man bei der Arbeit mit der Textkonsole durch verschiedene Tricks viel Aufwand und Zeit sparen kann. Deshalb soll es hier vor allem um die Textkonsole gehen.

Erreichen kann man die Textkonsole von der graphischen Oberfläche aus, indem man ein Terminalprogramm startet. Bei Gnome öffnet man dazu im Hauptmenü: `Anwendungen`→`Zubehör`→ `Terminal`. Alternativ kann man Alt - F2 drücken und in das Textfeld `gnome-terminal` eintippen. Bei anderen Oberflächen heißt das Terminal vielleicht anders (z. B. `konsole` oder `xterm`).

Loggt man sich per SSH-Client auf den Rechner ein, landet man bereits automatisch in einer Textkonsole. Man sieht eine Eingabeaufforderung (*prompt*).

Terminal
```
schueler@debian964:~$
```

Der Prompt beginnt mit dem Benutzernamen `"schueler"`; es folgen der Rechnername `"debian964"` und das aktuelle Verzeichnis `"~"`.

Das Programm, das sich mit der Eingabeaufforderung meldet, heißt *Kommandozeileninterpreter*. Bei Linux nennt man ein solches Programm *Shell*, weil es den Benutzer wie eine Muschel umgibt. Die bei Linux für Anwender am häufigsten benutzte Shell ist die *bash*[4]. Bei Windows wird stattdessen meistens das Programm `cmd.exe` benutzt.

Wie bei der Windows-Konsole kann man nun ein Programm oder einen Befehl starten, indem man den Programm- oder Befehlsnamen eingibt. Mit dem Drücken der Return- oder Enter-Taste wird der Befehl abgeschickt.

Das Programm oder der Befehl starten dann. Je nachdem, was man gestartet hat, kann man nun auf der Konsole sehen, was das Programm ausgibt; möglicherweise wird man zu einer Eingabe aufgefordert. Nach dem Ende des Programms erscheint wieder die Eingabeaufforderung der Shell.

Worin liegt nun der Unterschied zwischen Programm und Befehl? Ein *Programm* liegt in einem bestimmten Verzeichnis eines Massenspeichers. Die Shell veranlasst, dass das Programm in einen freien RAM-Bereich geladen wird. Anschließend springt die CPU an die Startadresse des Programms[5].

Ein *Befehl* dagegen ist ein Unterprogrammabschnitt der (bereits laufenden) Shell selbst. Hier muss die Shell nichts ins RAM laden, sondern nur die CPU an die Startadresse des Unterprogrammabschnitts springen. Um den Unterschied deutlich zu machen, nennt man so einen Befehl auch *Shell-Befehl* oder *eingebauten Befehl*. Ob man nun ein Programm oder einen Befehl vor sich hat, ist in den meisten Fällen egal.

In den folgenden Beispielen sind die Benutzereingaben übrigens fett gedruckt, der Rest nicht.

Terminal
```
schueler@debian964:~$ date
Di 28. Mai 00:01:02 CEST 2024
```

Anders als bei Windows werden Großbuchstaben und Kleinbuchstaben bei der Shell *immer* unterschieden.

[4] Abkürzung für bourne again shell

[5] stark vereinfachte Darstellung, mehr darüber im Kapitel über Prozesse

Terminal
```
schueler@debian964:~$ DATE
bash: DATE: Kommando nicht gefunden.
```

Das Programm `date` gibt es, das Programm `DATE` offensichtlich nicht.

Anders als bei Windows spielen die Endungen der Dateinamen keine wesentliche Rolle. So werden Programme auch ohne die Endung auf `.exe` vom System erkannt.

Nach dem Befehls- oder Programmnamen kann man (wie bei Windows) noch weitere Worte, sogenannte *Parameter*, angeben, bevor man die Return-Taste drückt. Der Befehls- oder Programmname und auch die weiteren Worte müssen allerdings durch jeweils mindestens ein Leerzeichen (Tabulator geht auch) voneinander getrennt werden. Ob man ein Leerzeichen oder mehrere nimmt, ist egal. Das erste Wort der Befehlszeile ist immer der Befehls- oder Programmname[6].

Terminal
```
schueler@debian964:~$ factor 120 130
120: 2 2 2 3 5
130: 2 5 13
schueler@debian964:~$ date -u
Di 28. Mai 00:05:07 CEST 2024
```

Im ersten Beispiel (beim Programm `factor`) wurden die Primfaktoren der Zahlen ausgegeben, die als Parameter angegeben warenIm zweiten Beispiel (beim Programm `date`) hatte der Parameter als erstes Zeichen ein Minuszeichen. Ein oder zwei Minus- oder Pluszeichen am Anfang eines Parameters bedeuten, dass es sich bei diesem Parameter um eine sogenannte *Option* handelt[7]. Eine Option ist ein Parameter, bei dem dem Befehl ein Sonderwunsch mitgegeben wird. Im Beispiel wurde mit der Option `-u` der Befehl angewiesen. Das Programm gibt deshalb die Uhrzeit in UTC (Weltzeit) aus. Ob ein Programm seine Optionen wirklich an einem Minuszeichen erkennt, liegt allerdings in der Freiheit des Programmierers. Bei den eingebauten Befehlen wird es allerdings konsequent eingehalten.

Verlassen kann man Textkonsole immer über den Befehl `exit`[8].

1.2.3 Hilfe

Die Hilfe zu einem Programm (z. B. **`cal`**) bekommt man über das Programm **`man`**, das die entsprechende Hilfeseite (auf englisch *manual page*) ausgibt. Als Parameter gibt man den (oder die) Programmnamen an.

Terminal
```
schueler@debian964:~$ man date
DATE(1)           Dienstprogramme für Benutzer                DATE(1)

BEZEICHNUNG
       date - Ausgeben oder Setzen von Systemdatum und -zeit
...
```

Mit Q (wie *quit*) kann man die Hilfe wieder verlassen.

Die Hilfe zu einem eingebauten Befehl (z. B. **`cd`**) bekommt man über den eingebauten Befehl **`help`**. Auch hier gibt man als Parameter den Befehlsnamen an.

Terminal
```
schueler@debian964:~$ help time
time: time [-p] Pipeline
    Report time consumed by pipeline's execution.
...
```

[6] Ausnahme: Ein-/Ausgabe-Umleitungen dürfen auch am Anfang stehen.

[7] Bei Windows werden Optionen meist durch einen Schrägstrich eingeleitet.

[8] Oder über Strg + D am Zeilenanfang. Wenn man die Textkonsole per login betreten hat, kann man stattdessen auch `logout` nehmen.

1.2.4 Tastenkombinationen für die Textkonsole

Neben der Taste Q für das Beenden der Hilfe gibt es weitere hilfreiche Tastenkombinationen in der Textkonsole. In Tabelle 3 sieht man die wichtigsten von ihnen.

Tastenkombination	Bedeutung	Bemerkung
Q	Programmende	für Man-Pages und `less`
Strg + C	Programmabbruch	viele Konsolenprogramme
Strg + D	Ende einer Eingabedatei	für Programme, die eine Eingabe aus einer Datei erwarten
Strg + S	XOFF: Ausgabe anhalten	alle Konsolenprogramme
Strg + Q	XON: Ausgabe fortsetzen	alle Konsolenprogramme
⟷	Auto-Vervollständigung	bei Befehlseingabe in der Shell
↑	Zurückblättern in der History	bei Befehlseingabe in der Shell
↓	Vorblättern in der History	bei Befehlseingabe in der Shell
Shift ⇑ - Strg + C	Kopieren	nur in Gnome-Terminal
Shift ⇑ - Strg + V	Einfügen	nur in Gnome-Terminal

Tabelle 3: Wichtige Tastenkombinationen für die Textkonsole

1.2.5 Linux-Befehle zum Systemzustand

In Tabelle 4 sind einige erste Befehle und Programme aufgeführt; manche davon geben einen ersten Einblick zum Zustand des Systems.

Aktion	Art	Linux	Windows
Anzahl der Prozessorkerne anzeigen	P	**`nproc`**	—
Betriebssystem-Version anzeigen	P	**`uname -a`**	**`ver`**
Datum und Uhrzeit anzeigen	P	**`date`**	**`time`**
Uptime anzeigen	P	**`uptime`**	—
Liste aktueller Nutzer anzeigen	P	**`who`**	—
20 Sekunden warten	P	**`sleep 20`**	—
Text ausgeben	B	**`echo Hallo`**	**`echo Hallo`**
Hilfe zum Shell-Befehl x	B	**`help x`**	**`help x`**
Hilfe zum Programm x	P	**`man x`**	—
Shell verlassen	B	**`exit`**	**`exit`**

Tabelle 4: Auswahl einfacher Linux-Befehle

1.2.6 Dateien

Im Hauptspeicher findet man seine Daten (z.B. Variablen) anhand ihrer Adresse. So hat jede Variable in einem C++- oder Java-Programm eine eigene Speicheradresse. Innerhalb des Programmes wird diese Methode wesentlich vereinfacht durch Variablennamen. So fällt es dem Programmierer leichter, sich den Namen `radius` zu merken als die Adresse `0xb3b810`. Die Umsetzung zwischen Name und Adresse erledigt der Compiler.

Genauso hat es sich eingebürgert, Speicherbereiche auf Massenspeichern (Festplatten, DVDs usw.) mit eindeutigen Namen zu kennzeichnen. Diese Speicherbereiche heißen *Dateien*. Um die Größe oder gar die Lage der Dateien auf dem Massenspeicher braucht sich der Benutzer nicht zu kümmern. Das erledigt die Massenspeicher-Verwaltung des Betriebssystems.

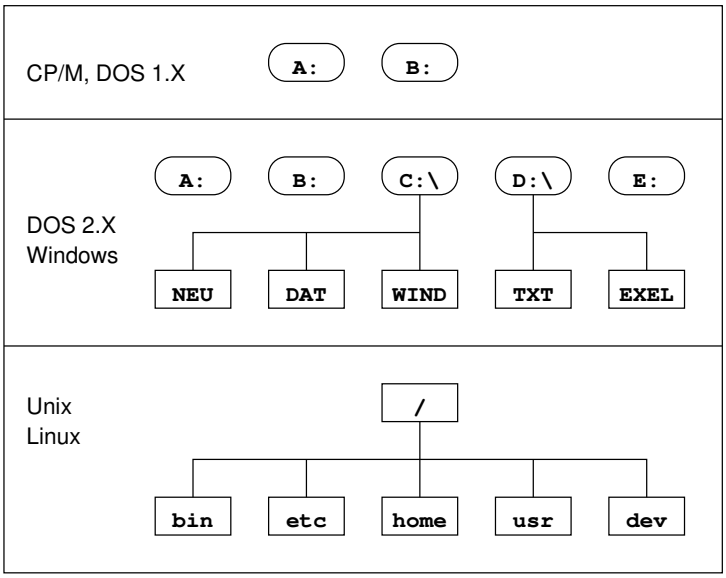

Abbildung 1: Verzeichnisbäume in verschiedenen Betriebssystemen

1.2.7 Verzeichnisse

Mit größer werdenden Massenspeichern reichen diese Namen nicht mehr aus. Erstens können leicht Doppelungen auftreten, zweitens wird eine Liste von beispielsweise 100000 Dateien schnell unübersichtlich. Die meisten Betriebssysteme ordnen daher den Namensraum weiter, indem sie Behälter für Dateien ermöglichen (*Verzeichnisse* oder *Ordner* genannt). Die Skalierbarkeit liegt darin, dass jedes Verzeichnis sowohl Dateien als auch weitere Verzeichnisse enthalten kann. Es entsteht ein baumförmiges System, der sogenannte Verzeichnisbaum[9]. Auf diese Art erweitert sich der ursprüngliche Dateiname um die Verzeichnisnamen; er wird damit zum Pfadnamen (siehe unten).

1.2.8 Unterschied 1: Laufwerksbuchstaben

Bei Linux sieht die Massenspeicher-Organisation (=Dateisystem) nun so aus, dass der Verzeichnisbaum genau *eine* Wurzel haben kann, von dem aus alle weiteren Verzeichnisse und alle Dateien ausgehen.

Im Gegensatz dazu hat bei Windows jeder Massenspeicher einen eigenen unabhängigen Verzeichnisbaum, beginnend mit einem sogenannten Laufwerksbuchstaben.

Der Vorteil des Windows-Verfahrens liegt in der Abwärts-Kompatibilität zu MS-DOS, der Vorteil des Linux-Verfahrens liegt darin, dass durch Auf- oder Abwärtsgehen im Baum jede Stelle des Baums erreicht werden kann. Abbildung 1 stellt die Unterschiede noch einmal dar.

Wird unter Linux ein neuer Massenspeicher installiert, so wird er vom System an eine Stelle des Verzeichnisbaums, nämlich an ein (meist leeres) Verzeichnis montiert, er wird gemountet (mittlerweile wird diese Technik auch bei Windows-Systemen verwendet). Bei Benutzung von Wechseldatenträgern wird dieser Prozess heute meist automatisiert.

1.2.9 Unterschied 2: Pfadnamentrenner

Für den Zugriff auf eine Datei in einem Verzeichnisbaum braucht man einen eindeutigen Namen. Er setzt sich aus den Namen der betroffenen Verzeichnisse und dem Dateinamen selbst zusammen

[9]Manchmal auch Dateisystem genannt; aber dieser Begriff wird auch für die verschiedenen Formate der technischen Realisierung eines solchen Verzeichnisbaums benutzt, wie z. B. EXT4 oder FAT32.

und heißt *absoluter Pfadname*. Durch diese Festlegung darf es in verschiedenen Verzeichnissen verschiedene Dateien geben, die alle gleich heißen.

Der Pfadname setzt sich bei Linux wie folgt zusammen:

a) Er beginnt mit dem Namen des Wurzelverzeichnisses, nämlich `"/"`

b) Danach folgen, falls vorhanden, in der Reihenfolge die Namen der Verzeichnisse, die man im Baumgraphen durchläuft, um zur Datei zu kommen; getrennt werden sie ebenfalls durch das Trennzeichen `"/"`.

c) Zum Schluss kommt der Dateiname, ebenfalls durch das Trennzeichen `"/"` abgetrennt.

d) Bei einem Verzeichnis darf man ein `"/"` an den Namen anhängen, um deutlich zu machen, dass es sich um ein Verzeichnis handelt.

Die mit X markierte Datei `neu.txt` in Abbildung 2 hat dann den Pfadnamen `/home/omi/neu.txt`.

Der Unterschied zwischen den Betriebssystemen liegt *nur* im benutzten Trennzeichen: Linux benutzt den Schrägstrich `"/"`, der Pfadname lautet `/home/omi/neu.txt`. Windows benutzt den Backslash `"\"`, hier lautet er `\home\omi\neu.txt`.

Das Zeichen `"/"` bei Linux (bzw. `"\"` bei Windows) hat also – je nach Ort im Pfadnamen – drei verschiedene Bedeutungen:

a) Ganz vorne steht es für das Wurzelverzeichnis (=Stammverzeichnis)

b) In der Mitte dient es als Pfadnamentrenner.

c) Am Schluss kann es angefügt werden, um zu sagen: der Pfadname ist der Name eines Verzeichnisses.

1.2.10 Unterschied 3: Groß- und Kleinschreibung

Weiterhin unterscheidet Linux bei allen Namen, also auch bei Dateinamen und Verzeichnisnamen zwischen Groß- und Kleinschreibung.

1.2.11 Unterschied 4: Versteckte Dateien und Verzeichnisse

Bei Linux gelten Dateien und Verzeichnisse, deren Namen mit einem Punkt beginnen, als versteckt. Das soll nicht der Sicherheit dienen, sondern der Bequemlichkeit. So gibt es im persönlichen Verzeichnis von Benutzern häufig Konfigurations- und Backup-Dateien. Sie sind zwar wichtig, aber der Benutzer will sie nicht ständig sehen und beachten müssen. Deshalb werden sie von den üblichen Befehlen und Programm nicht beachtet und angezeigt. Meistens gibt es aber eine Option (oft `-a` oder `--all`), die dieses Verhalten abschaltet.

1.2.12 Ein typischer Linux-Verzeichnisbaum

Abbildung 2 zeigt einen für Linux typischen Verzeichnisbaum. In diesem Baum hat jedes Verzeichnis (bzw. der darunterliegende Teilbaum) einen bestimmten Zweck, wie man aus Tabelle 5 entnehmen kann. Im sogenannten *Filesystem Hierarchy Standard*, abgekürzt FHS, sind weitere Einzelheiten festgelegt. Einen kurzen Überblick bekommt man, indem man `man hier` aufruft (`hier` steht für Hierarchy).

1.2.13 Aktuelles Verzeichnis

Standardmäßig beziehen sich viele Programme und Befehle auf das aktuelle Verzeichnis. Das ist das Verzeichnis, das meist als letzter Teil des Prompts dargestellt wird. So ist es auch beim Programm `ls` (Abkürzung für *list*). Dieses Programm listet die Namen aller Dateien und Verzeichnisse in einem Verzeichnis auf.

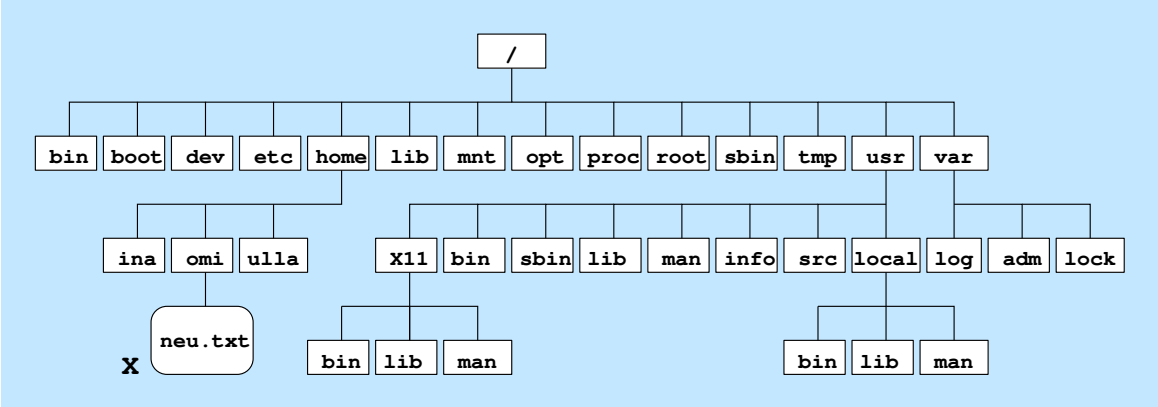

Abbildung 2: Linux-Verzeichnisbaum

Teilbaum	Zweck
`/bin, /sbin`	Programme für den Administrator
`/dev`	Gerätedateien (Drucker usw.)
`/etc`	Konfigurationsdateien
`/home`	Persönliche Verzeichnisse
`/lib`	DLLs und statische Libraries
`/mnt`	beliebter Mount-Punkt
`/proc`	aktuelle Systemdaten in Dateiform
`/opt`	selbst installierte Pakete
`/root`	Heimatverzeichnis des Administrators `root`
`/tmp`	temporäre Dateien
`/usr/bin`	Anwenderprogramme für Normalbenutzer
`/usr/local`	Anwenderprogramme, selbst installiert
`/usr/man`	Hilfe-System
`/var`	Systemdateien zum Loggen und Spoolen

Tabelle 5: Zweck der einzelnen Verzeichnisse

Terminal
```
schueler@debian964:~$ ls # zeigt den Inhalt des aktuellen Verz. an
schueler@debian964:~$ ls /usr/lib # zeigt den Inhalt von /usr/lib an
```

Unmittelbar nach dem Einloggen ist das Heimatverzeichnis (meist `/home/nutzername` oder `/home/gruppe/nutzername`) das aktuelle Verzeichnis. Ermitteln kann man den Namen des aktuellen Verzeichnisses mit dem Befehl `pwd` (=*print working directory*):

Terminal
```
schueler@debian964:~$ pwd
/home/schueler
```

Der Benutzer kann sein aktuelles Verzeichnis jederzeit selbst ändern (soweit die Verzeichnisrechte es erlauben). Dazu gibt es den Befehl `cd` (=*change directory*).

Terminal
```
schueler@debian964:~$ cd /etc   # wechseln nach /etc
schueler@debian964:/etc$ pwd
/etc
schueler@debian964:/etc$ cd # ohne Param.:zurueck ins pers.Verz.
schueler@debian964:~$ pwd
/home/schueler
schueler@debian964:~$ cd ..     # jetzt im Baum eine Ebene hoch
schueler@debian964:/home$ pwd
/home
```

1.2.14 Relativer Pfadname

Wenn man das aktuelle Verzeichnis kennt, kann man sich das Leben vereinfachen, indem man *relative Pfadnamen* für Dateien und Verzeichnisse verwendet. Relative Pfadnamen zählen nicht ab dem Wurzelverzeichnis `/`, sondern ab dem aktuellen Verzeichnis.

Ist man z.B. in `/home/omi` und möchte `/home/omi/neu.txt` löschen, dann kann man schreiben:

Terminal
```
schueler@debian964:/home/omi$ pwd
/home/omi
schueler@debian964:/home/omi$ rm neu.txt # statt rm /home/omi/neu.txt
```

`neu.txt` ist ein relativer Pfadname für die gemeinte Datei, `/home/omi/neu.txt` ist der absolute Pfadname für dieselbe Datei. Man muss jedoch dabei aufpassen, dass man sich im richtigen Verzeichnis befindet und die richtige Datei löscht!

Absolute Pfadnamen erkennt man allein daran, dass sie mit einem Slash (`/`, Schrägstrich) beginnen. Bei relativen Pfadnamen fehlt dieser Slash am Anfang.

Relative Pfadnamen dürfen auch selbst Verzeichnisnamen enthalten:

Terminal
```
schueler@debian964:~$ cd /home
schueler@debian964:/home$ rm omi/neu.txt # D.im Verz. unterhalb
schueler@debian964:/home$ mkdir omi/neuordner # Unterverz. anlegen
```

1.2.15 Abkürzungen für aktuelles Verzeichnis und Oberverzeichnis

Damit man bei relativen Pfadnamen auch das eigene Verzeichnis sowie das Oberverzeichnis ansprechen kann, gibt es zwei Abkürzungen: Der Verzeichnisname `..` (zwei Punkte) bezeichnet das Oberverzeichnis des aktuellen Verzeichnisses. Somit kann man mit `cd ..` eine Ebene höher wechseln. Der Verzeichnisname `.` (ein Punkt) bezeichnet das aktuelle Verzeichnis selbst. Mit `cd .` bleibt man daher im gleichen Verzeichnis.

Terminal
```
schueler@debian964:/home/omi/neuordner$ pwd
/home/omi/neuordner
schueler@debian964:/home/omi/neuordner$ rm ../neu.txt #oberhalb
schueler@debian964:/home/omi/neuordner$ mkdir ../../ina/neuordner
```

Die Shell versteht noch eine weitere Abkürzung, die nichts mit dem aktuellen Verzeichnis zu tun hat. Und zwar bezeichnet ~ (Tilde) das Heimatverzeichnis des Benutzers.

1.2.16 Linux-Befehle für Dateien

Tabelle 6 zeigt nun eine Reihe wichtiger Programme und Befehle für den Umgang mit Dateien. An den Stellen der Tabelle, an denen drei Punkte stehen, ist aus Platzgründen jeweils nur ein

Aktion	Befehlszeile
Alle Dateinamen[10] anzeigen	**ls**
Mit Eigenschaften	**ls -l...**
Leere Datei anlegen	**touch datei ...**
Datei kopieren	**cp quelldatei zieldatei**
Datei umbenennen	**mv quelldatei zieldatei**
Datei löschen	**rm datei ...**
Datei-Inhalt bearbeiten	**nano datei**
Datei-Inhalt anzeigen	**cat datei ...**
Seitenweise anzeigen	**less datei ...**
Datei-Inhalt drucken	**lpr datei ...**
Inhaltstyp nennen	**file datei ...**

Tabelle 6: Auswahl von Befehlen für Dateien

Dateiname angegeben, obwohl beliebig viele möglich sind.

1.2.17 Linux-Befehle für Verzeichnisse

Tabelle 7 zeigt nun eine Reihe wichtiger Programme und Befehle für den Umgang mit Verzeichnissen. An den Stellen der Tabelle, an denen drei Punkte stehen, ist aus Platzgründen jeweils nur ein Datei- oder Verzeichnisname angegeben, obwohl beliebig viele möglich sind. Beim Umbenennen eines Verzeichnisses muss man darauf achten, dass der neue Verzeichnisname noch nicht existiert; falls doch, wird das Verzeichnis in das Zielverzeichnis verschoben.

1.2.18 Editoren

Soll eine Textdatei erstellt werden, kann dies am Anfang am besten mit einem geeigneten Editor geschehen. Tabelle 8 zeigt ein paar von ihnen.

Der Editor **nano** ist für Anfänger, die im Textmodus arbeiten, zu empfehlen. Er zeigt einige seiner Funktionstasten in den beiden Fußzeilen an (je breiter das Fenster ist, desto mehr Funktionstasten werden angezeigt):

Terminal
```
^G Hilfe   ^O Speichern    ^W Wo ist   ... M-U Rückgängig
^X Beenden ^R Datei öffnen ^ Ersetzen ... M-E Wiederholen
```

Dabei bedeutet ^ die Strg -Taste und M- die Alt -Taste:

- ^G bedeutet Strg + G
- M-U bedeutet Alt + U

Aktion	Befehlszeile
Verzeichnisinhalt anzeigen	**`ls verzeichnis`**
mit Einzelheiten	**`ls -l everzeichnis`**
mit versteckten Objekten	**`ls -a verzeichnis`**
nicht Inhalt, sondern Verz. selbst	**`ls -a verzeichnis`**
Aktuelles Verzeichnis wechseln	**`cd zielverzeichnis`**
Ein Verzeichnis nach oben	**`cd ..`** (mit Leerzeichen!)
Ins persönliche Verzeichnis	**`cd`**
Name des aktuellen Verzeichnisses	**`pwd`**
Verzeichnis erstellen	**`mkdir verzeichnis`**
Verzeichnis löschen	**`rmdir verzeichnis`**
Verzeichnis löschen mit Inhalt	**`rm -r verzeichnis`**
Datei kopieren in Verzeichnis	**`cp datei ...verzeichnis/`**
Datei verschieben in Verzeichnis	**`mv datei ...verzeichnis/`**
Verzeichnis umbenennen	**`mv verzeichnis zielverzeichnis`**
Verzeichnis in anderes verschieben	**`mv quellverzeichnis ...zielverzeichnis/`**
Verzeichnis kopieren	**`cp -r quellverzeichnis ...zielverzeichnis/`**

Tabelle 7: Auswahl von Linux-Programmen und -Befehlen für Verzeichnisse

Name	Modus	Bemerkung
`joe`	Text	empfehlenswert, Hilfe mit Strg + K - H
`nano`	Text	einfacher Editor; anderer Name: **`pico`**
`vi`	Text	klein und überall vorhanden, erfordert Einarbeitung
`emacs`	Text	unbegrenzt erweiter- und anpassbar
`gedit`	Graphik	einfach zu bedienen
`geany`	Graphik	einfach zu bedienen, als Kleinst-IDE verwendbar

Tabelle 8: Auswahl von Editoren

1.2.19 Namen mit Leer- und Satzzeichen (Maskierung)

Wenn man zwei Verzeichnisse mit den Namen `VW` und `Opel` anlegen möchte, tippt man `mkdir VW Opel` ein, und schon werden die zwei Verzeichnisse angelegt:

Terminal
```
schueler@debian964:~$ mkdir VW Opel
schueler@debian964:~$ ls -l
...
drwxr-xr-x 2 schueler schueler 4096 28. Mai 21:00 VW
drwxr-xr-x 2 schueler schueler 4096 28. Mai 21:00 Opel
```

Was ist aber, wenn man nur *ein* Verzeichnis anlegen möchte, das den Namen `Skoda Fabia` hat? Mit der Zeile `mkdir Skoda Fabia` passiert dies:

Terminal
```
schueler@debian964:~$ mkdir Skoda Fabia
schueler@debian964:~$ ls -l
...
drwxr-xr-x 2 schueler schueler 4096 28. Mai 21:01 Skoda
drwxr-xr-x 2 schueler schueler 4096 28. Mai 21:01 Fabia
```

Weil der gewünschte Name ein Leerzeichen hat, wird er durch die Shell (den Befehlsinterpreter der Konsole) als zwei Namen erkannt. Es wird nämlich anhand der Leer- und Tabulatorzeichen eine *Worttrennung* ausgeführt. Das ist nötig, damit man Befehl und Parameter und die Parameter untereinander auseinanderhalten kann.

Um diese und andere Aktionen der Shell zu vermeiden, muss man bei solchen Namen eine *Maskierung* solcher Sonderzeichen vornehmen. Maskieren heißt hier: Die Sonderzeichen innerhalb eines Namens werden unwirksam gemacht.

Dazu hat man drei Möglichkeiten:

a) `Skoda\ Fabia`: Jedes Sonderzeichen wird durch einen davor stehenden Backslash maskiert.

b) `'Skoda Fabia'`: Der Name wird in Hochkommata gesetzt.

c) `"Skoda Fabia"`: Der Name wird in Anführungszeichen gesetzt.

Alle drei Möglichkeiten sind gleichwertig. Allerdings werden bei der dritten Möglichkeit das Dollarzeichen (`$`) und der Backtick (`` ` ``) nicht maskiert (dazu später). Maskiert sieht unsere Zeile also so aus:

Terminal
```
schueler@debian964:~$ mkdir "Skoda Fabia"
schueler@debian964:~$ ls -l
...
drwxr-xr-x 2 schueler schueler 4096 28. Mai 21:01 'Skoda Fabia'
```

Die Hochkommata um den Namen bei der Ausgabe des `ls`-Befehls sind nur eine neue „Errungenschaft“ einiger neuer Versionen von `ls`. Der Name des Verzeichnisses hat keine Hochkommata (wir haben ja auch keine eingegeben!).

Welche Zeichen sollte man nun maskieren und welche nicht?

- Problemlos sind alle Buchstaben und Ziffern, der Unterstrich, Punkt, Plus- und Minuszeichen. Falls der Name nur solche enthält, ist die Maskierung überflüssig. Es ist klug, auf Umlaute und Akzente zu verzichten.
- Schrägstrich und der Terminator `'\0'` können nicht in Namen vorkommen – da hilft auch keine Maskierung.
- Maskieren muss man: Leerzeichen, Tabulator, Zeilenumbruch, alle nicht genannten Satz- und Sonderzeichen, besonders das Dollarzeichen, das Fragezeichen und Stern.

1.2.20 Namen, die wie Optionen heißen

Ein Sonderfall kann nun trotzdem noch auftreten: Jemand kann auf die Idee kommen, seine Datei oder sein Verzeichnis so zu nennen wie eine Option. Zum Beispiel möchte jemand ein Verzeichnis mit dem Namen `--help` anlegen:

Terminal

```
schueler@debian964:~$ mkdir --help
Aufruf: mkdir [OPTION]... VERZEICHNIS...
Erzeugen der/des Verzeichnisse(s), wenn sie noch nicht existieren.
...
schueler@debian964:~$ ls -d --help
Aufruf: ls [OPTION]... [DATEI]...
Auflistung von Informationen über die DATEIen (Vorgabe ist das
aktuelle Verzeichnis).
...
```

Hier haben beide Befehle versagt: Sie haben `--help` für eine Option gehalten. Dieses Problem kann man auch durch Maskierung nicht lösen, denn es sind die Programme selbst, die den Namen falsch (nämlich als Option) verstehen.

Deshalb hat das Programm `mkdir` (und viele andere auch) eine Sonderoption, die ihm sagt, ab wo Schluss ist mit den Optionen. Die Sonderoption `--` sagt: Ich bin die letzte Option, nach mir stehen nur noch „normale“ Parameter auf der Befehlszeile:

Terminal

```
schueler@debian964:~$ mkdir -- --help
schueler@debian964:~$ ls -l -d -- --help
drwxr-xr-x 2 schueler schueler 4096 28. Mai 22:02 --help
```

Der `ls`-Befehl im Beispiel hat also drei Optionen: `-l` ist für die ausführliche Ausgabe; `-d` sorgt dafür, dass das Verzeichnis nur von außen angesehen wird; `--` ist das Zeichen dafür, dass nun keine Optionen mehr kommen. Deshalb wird `--help` nicht als Option angesehen, sondern als Verzeichnisname.