# Vorlesung Linux-Praktikum

## 1. Einführung: Dateisystem und erste Schritte

Dirk Frettlöh

Technische Fakultät
Universität Bielefeld

# Erste Schritte mit der Kommandozeile

## Aufgaben der Kommandozeile

1. Programme ausführen
2. Programme zu mächtigeren Werkzeugen kombinieren (`for i in *jpg ... convert $i`)
3. Kommandozeilen-Skripte
   - 1) und 2) abstrahieren und in Datei speichern
   - wiederverwenden statt erneut eintippen

Zu 2 und 3 später. Zu 1:

# Programme aufrufen etc

# Erste Schritte mit der Kommandozeile

## Programme in der Kommandozeile aufrufen

(Erinnerung: Kommandozeile = Shell = Terminal: öffne das Programm "Terminal")

Auf einer Linuxkiste: in einer Shell z.B.

- `$ firefox`
- `$ libreoffice form.docx`
- `$ date`
- `$ ls ordner/`

Das ist eine der wenigen Stellen in dieser Vorlesung, die auf Windowskisten und Mackisten anders ist. Da klappt obiges nur für Programme, die in der shell selbst laufen (ls, date), nicht für welche mit eigenem Fenster (Firefox, Libreoffice). Bei Mac geht folgender Umweg:

```
$ open /Application/Firefox.app
```

# Erste Schritte mit der Kommandozeile

## Schreib- und Sprechkonventionen



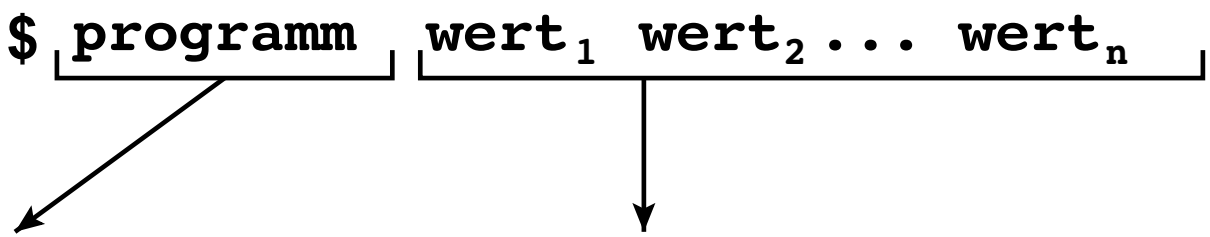
Schreibweise:


Sprechweise:

- Programme ausführen / aufrufen
- Dateien (mit einem Programm) öffnen

# Erste Schritte mit der Kommandozeile

## Programmaufruf allgemein




**Programmname:**
– immer an erster Stelle
– Name muß eindeutig sein

**Aufruf–Werte:**
- durch Leerzeichen getrennt
– in Anführungszeichen "als ein Wort"
– Interpretation vom Programm abhängig

Nützlich: “Pfeil-hoch”-Taste (↑) blättert durch die letzten eingegebenen Befehle.

# Erste Schritte mit der Kommandozeile

## Tab-Vervollständigung

Sehr nützlich:

**Tab-Vervollständigung**: Nur den Anfang eines Befehls eingeben, dann die Tab-Taste:

- Falls es nur eine mögliche Fortsetzung gibt, wird das Wort vervollstandigt Z.B. `libr` [Tab] wird zu `libreoffice`.
- Falls nicht, dann nicht. Aber:
- Falls nicht, dann: zweimal hintereinander Tab liefert eine Liste der möglichen Vervollständigungen:
  Z.B `lib` [Tab] [Tab] liefert z.B.

```
libjingle-call  libreoffice  libpng12-config
libnetcfg       libtoolize
```

Klappt auch mit Dateinamen!

# Erste Schritte mit der Kommandozeile

## Aufruf mit zusätzlichen Werten



"Schalter"-Optionen": `-schalter`

- "schalter" aktivieren / durchführen

```
$ xclock -digital
```

Wertangabe: `-eigenschaft wert`

- nimmt angegebene Eigenschaft für "Wert"

```
$ xclock -bg blue
```

Kombinieren geht natürlich auch:

```
$ xclock -digital -bg blue
```

# Erste Schritte mit der Kommandozeile

## Aufrufmöglichkeiten herausfinden

### Hilfefunktion des Programms selbst:

```
$ programm -h
$ programm --help
```

### "Manual Pages"

```
$ man programm
```

Manual Pages können mehrere "Kapitel" haben :

`man 1 free` → Programm ‘‘free’’

`man 3 free` → Programmierung ‘‘free’’

Sonst:

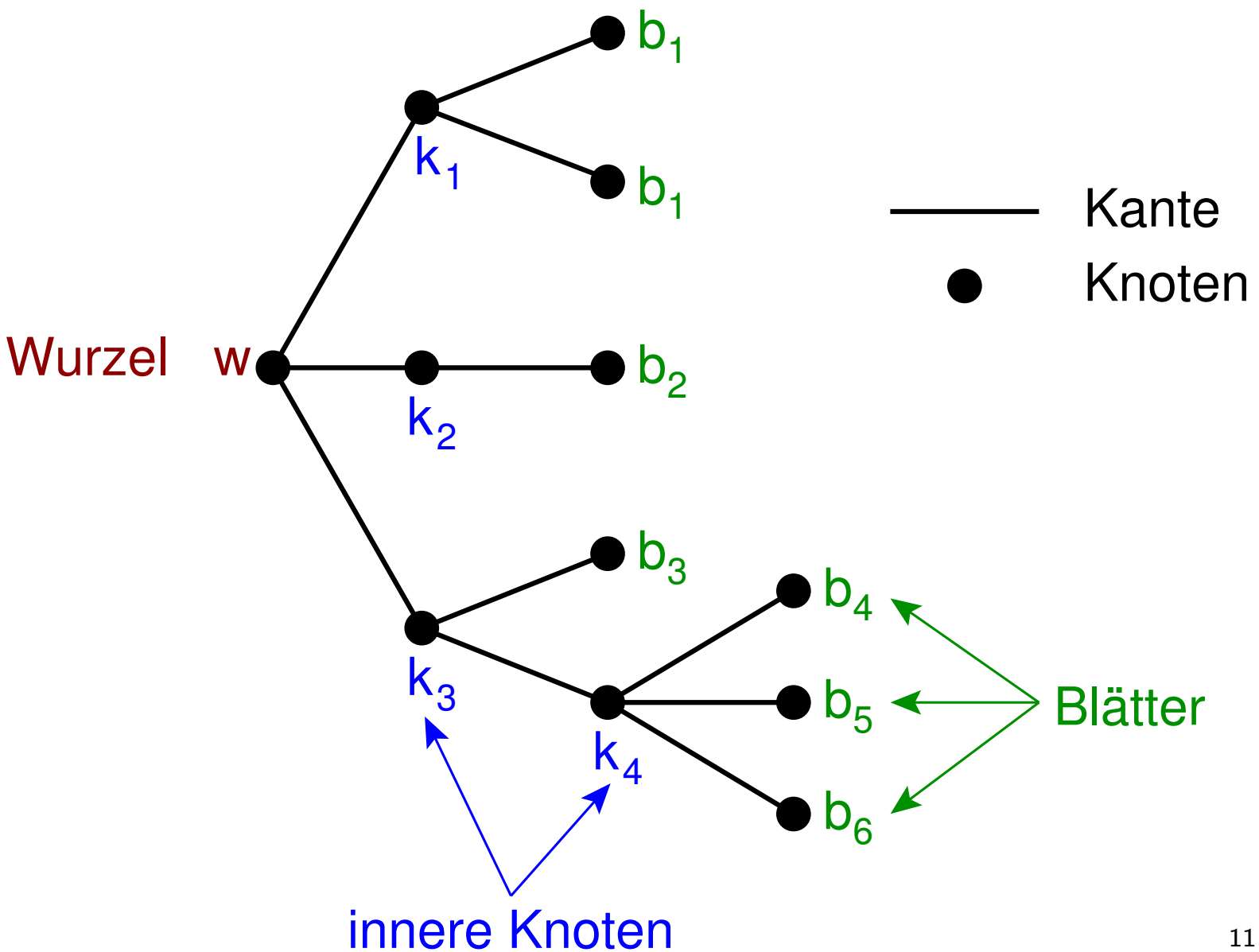
- `$ apropos stichwort` zeigt Befehle, die mit "stichwort" zu tun haben könnten
- Suchmaschine

# Unix-Dateisystem

# Dateisystem

## Bäume

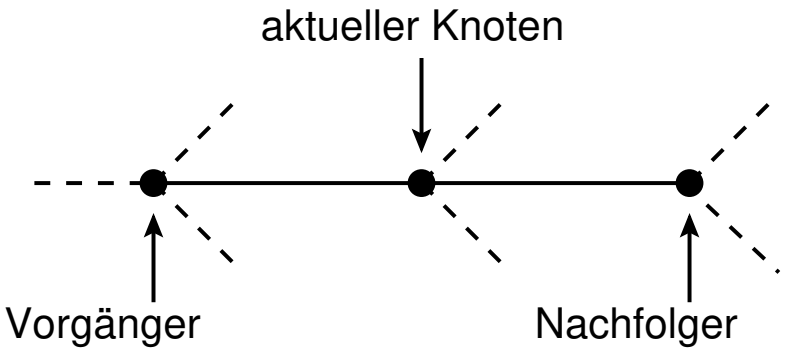


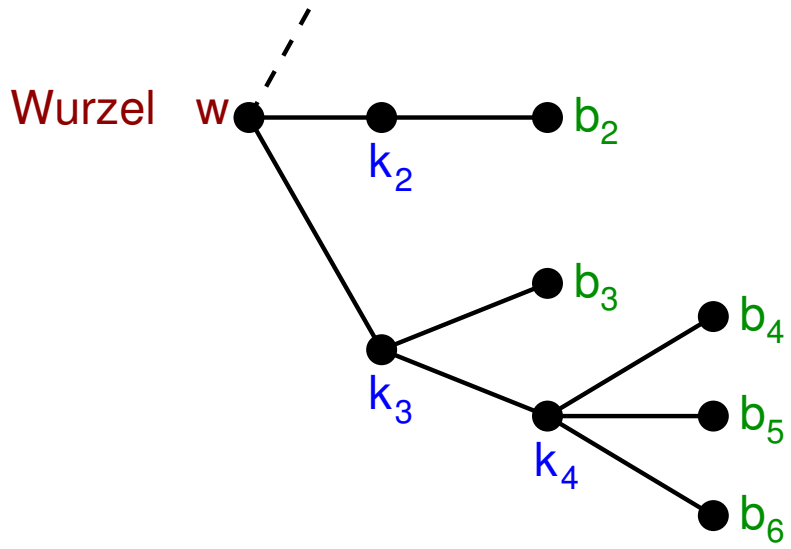


# Dateisystem

## Vorgänger/Nachfolger-Relation






Beispiel: $k_4$ hat Vorgänger $k_3$ und Nachfolger $b_4$, $b_5$, $b_6$.

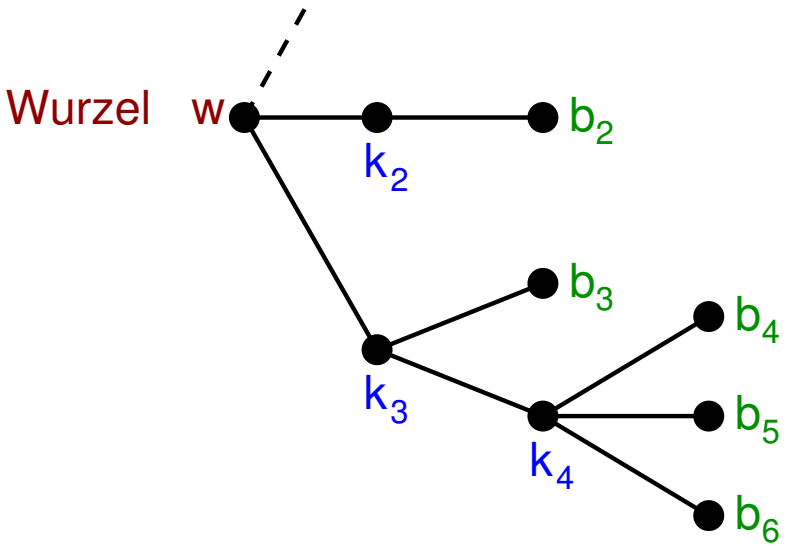
Daraus ergeben sich folgende Definitionen:

- Blätter haben keine Nachfolger.
- Die Wurzel ist der einzige Knoten ohne Vorgänger.
- Innere Knoten haben Vorgänger und Nachfolger.

# Dateisystem

## Pfade






Ein **Pfad** ist ein Weg von der Wurzel zu einem Knoten.

Notation: Aufschreiben der Knoten entlang des Pfades, z.B. $w/k_3/k_4/b_6$

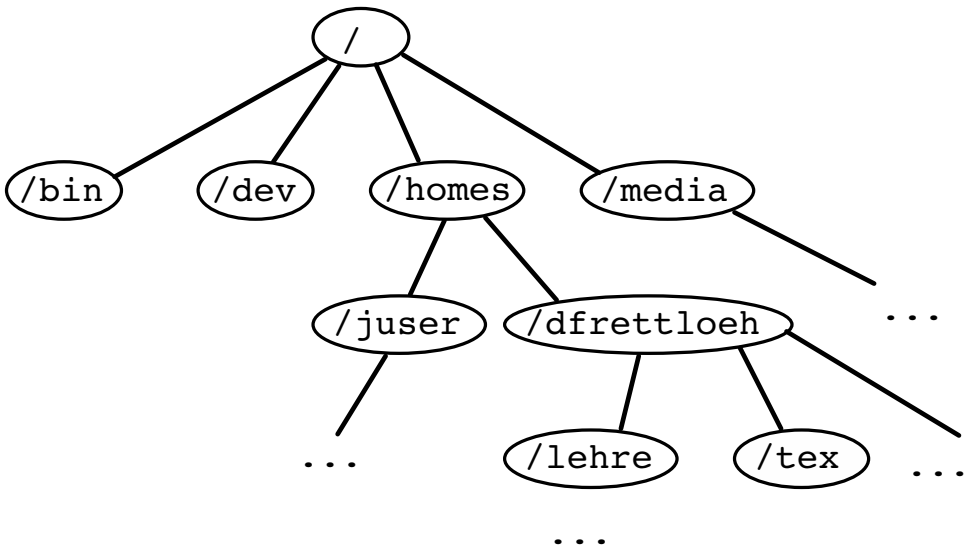
Zu jedem Knoten gibt es genau einen Pfad.

- Bäume haben keine Rundgänge ("Zyklen")

# Dateisystem

## Das Linux-Dateisystem ist ein Baum






- / : Wurzel
- Verzeichnisse: innere Knoten
- Dateien: Blätter

Pfade: /homes/dfrettloeh/lehre/unix/

Unix-Philosophie: alles ist eine Datei (z.B. USB-Stick, ...)

# Dateisystem

## Bewegen im Dateisystem

pwd (print working directory)

- zeigt momentane Position im Dateisystem
- genauer: den *Pfad* auf das Verzeichnis, in dem man sich gerade befindet.

```
$ pwd
/homes/dfrettloeh/lehre/unix/
```

# Dateisystem

## Bewegen im Dateisystem

### ls (list)

- zeigt Inhalt des aktuellen Verzeichnisses (ohne versteckte Dateien; vgl. nächste Folie)

```
$ ls
brief.odt
datei.txt
```

# Dateisystem

## Versteckte Dateien ("Punktdateien") anzeigen



- Dateien mit einem Punkt am Anfang sind versteckt (Beispiel: `.bashrc`)
- sieht man nur mit `ls -a`
- Verstecken ist nur Konvention zur Übersichtlichkeit, hat keine besondere Eigenschaft / Schutzfunktion!

```
$ ls -a
.punktdatei
brief.odt
datei.txt
```

# Dateisystem

## Zwei spezielle Punktdateien



. : Verweis auf das aktuelle Verzeichnis

```
$ ls .
```

.. : Verweis auf das Vorgänger-Verzeichnis

$\rightarrow$ wegen der Baumeigenschaft eindeutig!

```
$ ls ..
```

# Dateisystem

## in ein Unterverzeichnis wechseln

### cd (change directory)

- in ein anderes Verzeichnis wechseln

```
$ pwd
/homes/dfrettloeh/lehre/

$ cd unix

$ pwd
/homes/dfrettloeh/lehre/unix/
```

# Dateisystem

## in das Vorgängerverzeichnis wechseln



- `..` Verweis auf das Vorgängerverzeichnis (eindeutig; siehe Baumeigenschaft!)
- `..` wie normales Verzeichnis nutzbar

```
$ pwd
/homes/dfrettloeh/lehre/unix/

$ cd .

$ pwd
/homes/dfrettloeh/lehre/unix/

$ cd ..

$ pwd
/homes/dfrettloeh/lehre/
```

# Dateisystem

## in das Home des Nutzers wechseln



- Sonderfall: `cd` ohne Argument wechselt in das Home-Verzeichnis des Nutzers

```
$ pwd
/homes/dfrettloeh/lehre/unix/

$ cd

$ pwd
/homes/dfrettloeh
```

# Dateisystem

## Absolute Pfade

Kompletter Pfad von der Wurzel bis zum Ziel:

- wie normaler Datei-/Verzeichnisname verwendbar
- Vorteil: Man braucht nicht in das Zielverzeichnis zu wechseln, um dort etwas zu tun

```
$ pwd
/homes/dfrettloeh
```
*(aktuelles Verzeichnis: /homes/dfrettloeh !)*

```
$ libreoffice /homes/dfrettloeh/beispiele/brief.odt
```
*(öffnet Brief, der nicht im akt. Verzeichnis liegt)*

```
$ ls /homes/dfrettloeh/ablage
$ pwd
/homes/dfrettloeh
```
*(zeigt Inhalt von /homes/dfrettloeh/ablage, nicht des aktuellen Verzeichnisses!)*

```
$ cd /homes/dfrettloeh/beispiele
$ pwd
/homes/dfrettloeh/beispiele
```
*(wechselt in ein anderes Verzeichnis)*

# Dateisystem

## Relative Pfade

Pfad vom aktuellen Verzeichnis zum Ziel:

- wie normaler Datei-/Verzeichnisname verwendbar
- häufig kürzer als absoluter Pfad

```
$ pwd
/homes/dfrettloeh/beispiele/Bilder
```

```
$ cd ../..
```
*geht zwei Verzeichnisebenen zurück*

```
$ cd ../geschwister
```
*anderes Verz. auf gleicher Ebene*

```
$ cd ../eins/zwei
```
*eine Ebene hoch, dann zwei Ebenen tiefer*

# Dateisystem

## Dateien kopieren (im aktuellen Verzeichnis)

### cp (copy)

- kopiert eine Datei

```
$ cp brief.odt brief2.odt
```

# Dateisystem

## Dateien kopieren (in ein anderes Verzeichnis)

Die Kopie kann auch in einem anderen Verzeichnis liegen:

- mit dem gleichen Namen
- mit einem anderen Namen

```
$ pwd
/homes/dfrettloeh/beispiele/arbeit
$ cp brief.odt alt
$ cp brief.odt alt/peter.odt
```

# Dateisystem

## Unterverzeichnis anlegen

mkdir (make directory)

- legt ein Unterverzeichnis an

```
$ pwd
/homes/dfrettloeh/beispiele/arbeit

$ mkdir briefe
```

# Dateisystem

## Dateien/Verzeichnisse umbenennen

### mv (move)

- Datei / Verzeichnis umbenennen

```
$ pwd
/homes/dfrettloeh/beispiele/arbeit

$ mv datei.txt abc.txt
```

# Dateisystem

## Dateien/Verzeichnisse verschieben

Dateien und Verzeichnisse können auch in andere Verzeichnisse *verschoben* werden:

- und dabei ihren Namen behalten
- oder einen neuen Namen bekommen

```
$ pwd
/homes/dfrettloeh/beispiele/arbeit

$ mv datei.txt alt

$ mv datei.txt alt/xyz.txt
```

# Dateisystem

## Dateien löschen

### rm (remove)

- Datei löschen

```
$ rm datei
```

Vorsicht:

- Weg ist weg! Es gibt kein un-rm / undelete!
- Es gibt ein backup, aber das machen die RBG-Leute. Die sollten nur in wirklich wichtigen Fällen ins Spiel gebracht werden.

# Dateisystem

## Verzeichnisse löschen

### rmdir (remove directory)

- ein leeres Verzeichnis löschen

```
$ rmdir verzeichnis
```

### rm -rf (remove recursively)

- ein Verzeichnis mit allem Inhalt löschen
- Vorsicht!

# Dateisystem

## Wildcards

- dürfen als Bestandteile in Pfaden auftreten
  ($\rightarrow$ ls, mv, rm, ...)
- Stern `*` ersetzt beliebig viele Zeichen (auch 0):

  `k*.txt` passt auf `katalog.txt`, `kurs.txt`, `k2.txt`, und auch auf `k.txt`,
  aber nicht auf `kurs.doc` und `alkohol.txt`.
- Fragezeichen `?` ersetzt genau ein Zeichen:

  `aufg1?.txt` passt auf `aufg10.txt` und `aufg11.txt`,
  aber nicht auf `aufg1.txt` und `aufg101.txt`.

# Dateisystem

## Wildcards

- Liste [...] ersetzt genau ein Zeichen durch eines in der Liste

  `aufg1[123a].txt` passt auf `aufg11.txt` und `aufg1a.txt`,
  aber nicht auf `aufg10.txt` und `aufg17.txt`.

- Es geht auch [a-e] (=[abcde]) oder [3-6] (=[3456]) oder [A-E] (= [ABCDE])

# Textdateien

... und ihre Kodierung

# Dateitypen

## Dateien sind Bytefolgen

In der Hardware des Computers gibt es nur 0 und 1 (an/aus).

Früher: es konnten 8 Signale parallel transportiert und verarbeitet werden (heute 32 bzw 64).

Also konnte jeweils eine 8-stellige *Binärzahl* transportiert bzw verarbeitet werden, z.B. 10011010 oder 00111001.

So, wie ich eine Dezimalzahl wie 154 lese als

$$1 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0,$$

kann ich eine Binärzahl wie 10011010 lesen als

$$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^3 + 1 \cdot 2^1 + 0 \cdot 2^0.$$

Das ist ungewohnt, aber ebenfalls 154.

# Binärzahlen und Hexadezimalzahlen



Oder 01010100 als

$$0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0.$$

Also 84.

Wenn ich 16 Ziffern hätte statt 10 (*hexadezimal* statt dezimal), dann könnte ich eine 8-stellige Binärzahl mit nur zwei Ziffern schreiben.

Tun wir das: Ziffern 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
(A heißt zehn, B heißt elf, C heißt zwölf, ... F heißt fünfzehn.)

Um uns nicht zu vertun, schreiben wir Hexadezimalzahlen als 54h, B7h usw.

Dann ist $54\text{h} = 5 \cdot 16 + 4 \cdot 1 = 84$,
und $\text{B7h} = 11 \cdot 16 + 7 \cdot 1 = 183$.

# Dateitypen

## Dateien sind Bytefolgen



*Dezimal*

| | | | | | |
|---|---|---|---|---|---|
| ... | 84 | 101 | 120 | 116 | ... |

*Hexadezimal (Basis 16)*

| | | | | | |
|---|---|---|---|---|---|
| ... | 54h | 65h | 78h | 74h | ... |

# Dateitypen

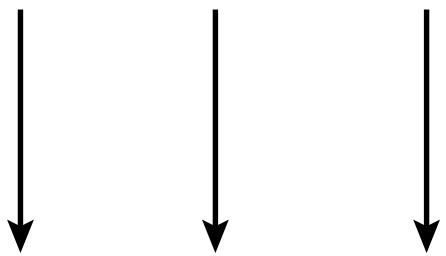
## ASCII-Kodierung

*Dezimal*

| ... | 84 | 101 | 120 | 116 | ... |
|---|---|---|---|---|---|

*Hexadezimal (Basis 16)*

| ... | 54h | 65h | 78h | 74h | ... |
|---|---|---|---|---|---|



*willkürliche (!) Abbildung von Bytes auf Buchstaben, Zeichen*

*ASCII*

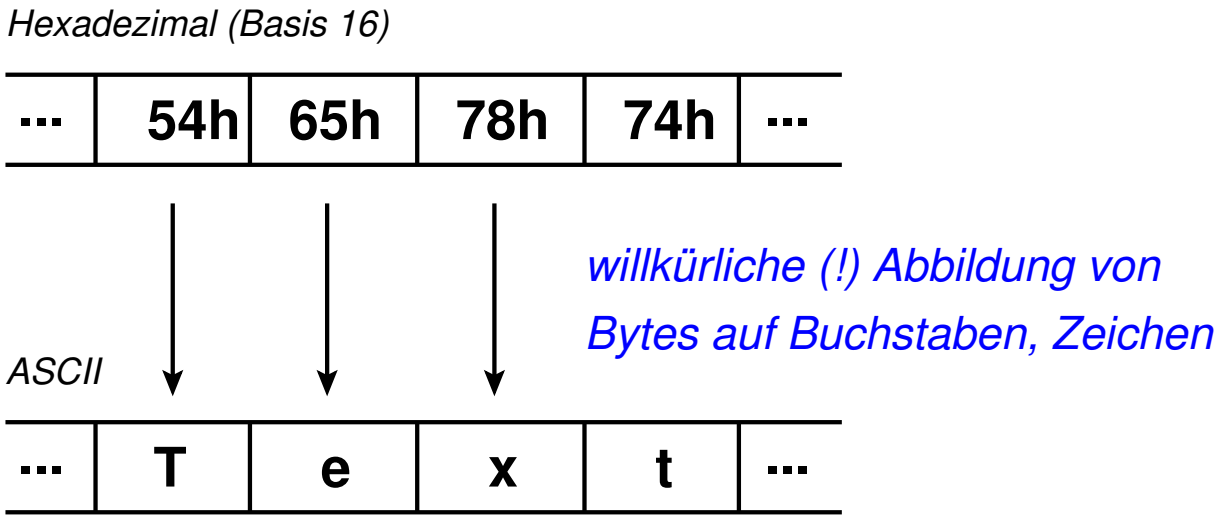
| ... | T | e | x | t | ... |
|---|---|---|---|---|---|

# Dateitypen

## Mit hexdump in die Datei hineinschauen

### hexdump (zeige Bytes einer Datei in Hexadezimal-Kodierung)

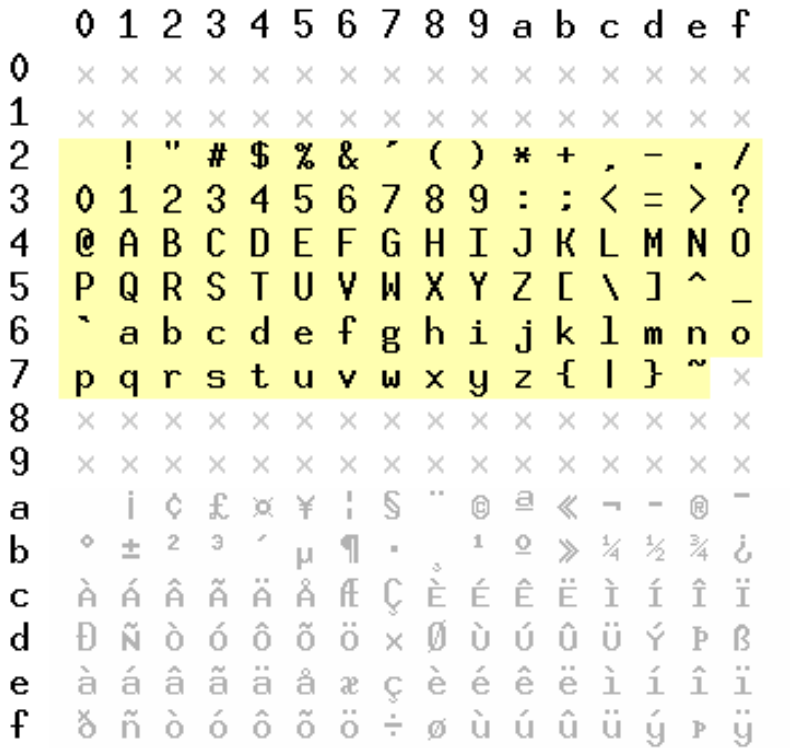
```
\$ hexdump -C test.txt
00000000  54 65 78 74 0a       |Text.|
00000005
```



# Dateitypen

## ASCII-Tabelle

**American Standard Code for Information Interchange**



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| 1 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| 2 | | ! | " | # | $ | % | & | ´ | ( | ) | * | + | , | − | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | × |
| 8 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| 9 | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × | × |
| a | | ¡ | ¢ | £ | ¤ | ¥ | ¦ | § | ¨ | © | ª | « | ¬ | - | ® | ¯ |
| b | ° | ± | ² | ³ | ´ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| c | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| d | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| e | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| f | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

**Textdatei:**
**nur die druckbaren Bytes**

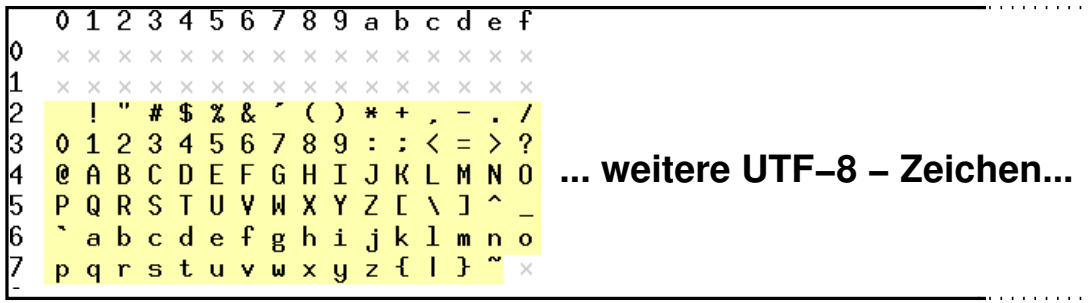
**Binärdatei:**
**alle beliebigen 256 Werte**

**(Tabelle: 16x16 = 256 Werte)**

# Dateitypen

## UTF-8-Kodierung

**UTF–8: Moderne Zeichenkodierung mit bis zu 4 Bytes**




**... weitere UTF–8 – Zeichen...**

- 7-Bit-ASCII ist gültiges UTF-8
  - (z.B. binär 0110 0101 = 65h = “e”)
- Achtes Bit =1: dann lies die nächsten Bytes.
  - Z.B. binär 11001000 100111110 = “Ȟ”
  - oder binär 11100000 10100110 10010101 = “ক” usw.
- Theoretisch bis zu 8 Bytes, vier Billionen Zeichen
- Real umgesetzt 4 Bytes, 1 114 112 Zeichen

° ® ™ ½ ¼ µ ¢ £ € ♠ ♣ Ǯ Đ Щ Ъ आ इ ر ذ د خ ⠿ ⠿ ⠿ ⠿ ⠿ ⠿
僑 僿 債 　 　 叉 毋 比 毛

# Dateitypen

## UTF-8-Kodierung

**Textdateien** ≠ **Dokumente**

```
Dokumente sind
keine Textdateien!

Sie sind
* Binärdateien oder wie
* Programmiersprachen
   aufgebaut.
```

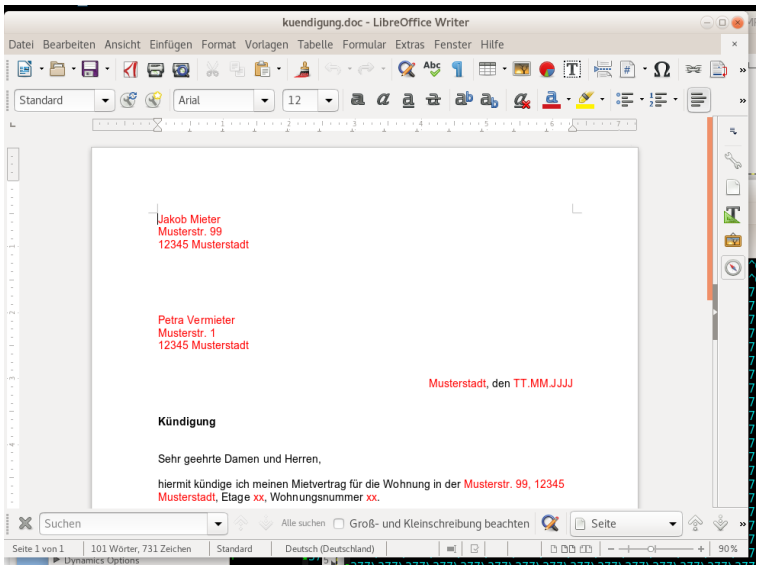
Dokumente sind
keine Textdateien!

Sie sind
- **Binärdateien** oder wie
- *Programmiersprachen*
  aufgebaut.

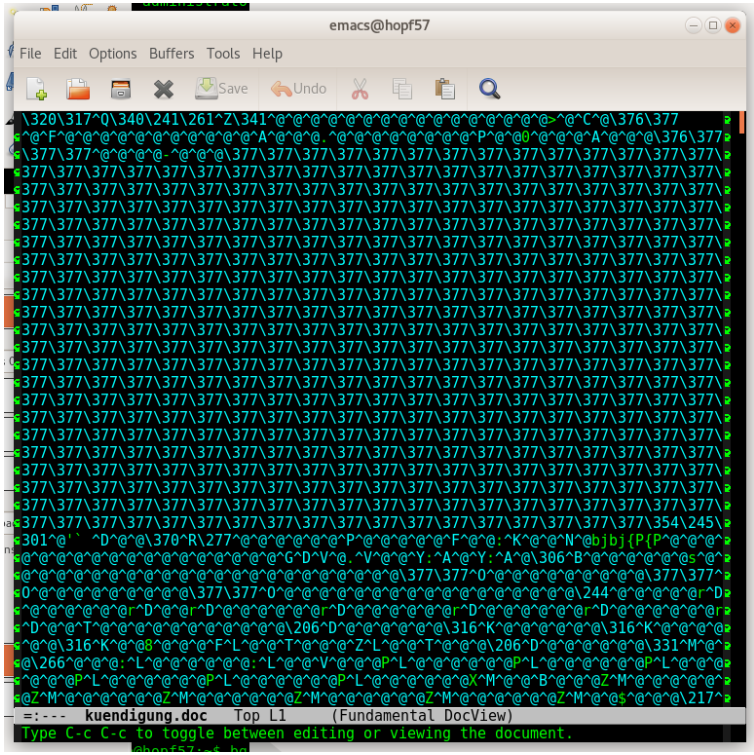
# Dateitypen

## Texteditoren und Textverarbeitung






Das zeigt eine Officesoftware




Das steht wirklich in der Datei.

# Dateitypen

## Beispiele für Textdateien



- Quellkode von Programmen (.c, .java-Dateien)
- Konfigurationsdateien (.bashrc, system.ini)
- Shellskripte (skript.sh, skript.bat)
- Ein-/Ausgaben von Kommandozeilen-Programmen

Wir arbeiten fast ausschließlich mit Textdateien.

Bitte für Programmcode, Shellskripte... nie Office-Programme benutzen.

Texteditoren: nano, emacs, vim, gedit... (für alles)
Notepad++, Eclipse... (speziell zum Programmieren)

# Dateitypen

## Textdateien betrachten

more

- Anzeigen, Blättern, Suchen in Textdateien

```
$ more textdatei
```

| | |
|---|---|
| [Leertaste] | eine Seite nach unten |
| b | eine Seite nach oben |
| [Return] | eine Zeile nach unten |
| y | eine Zeile nach oben |
| /suchbegriff | nach einem Begriff suchen |
| n | Suche fortsetzen |
| h | eingebaute Hilfe zu more |

# Dateitypen

## Texteditoren

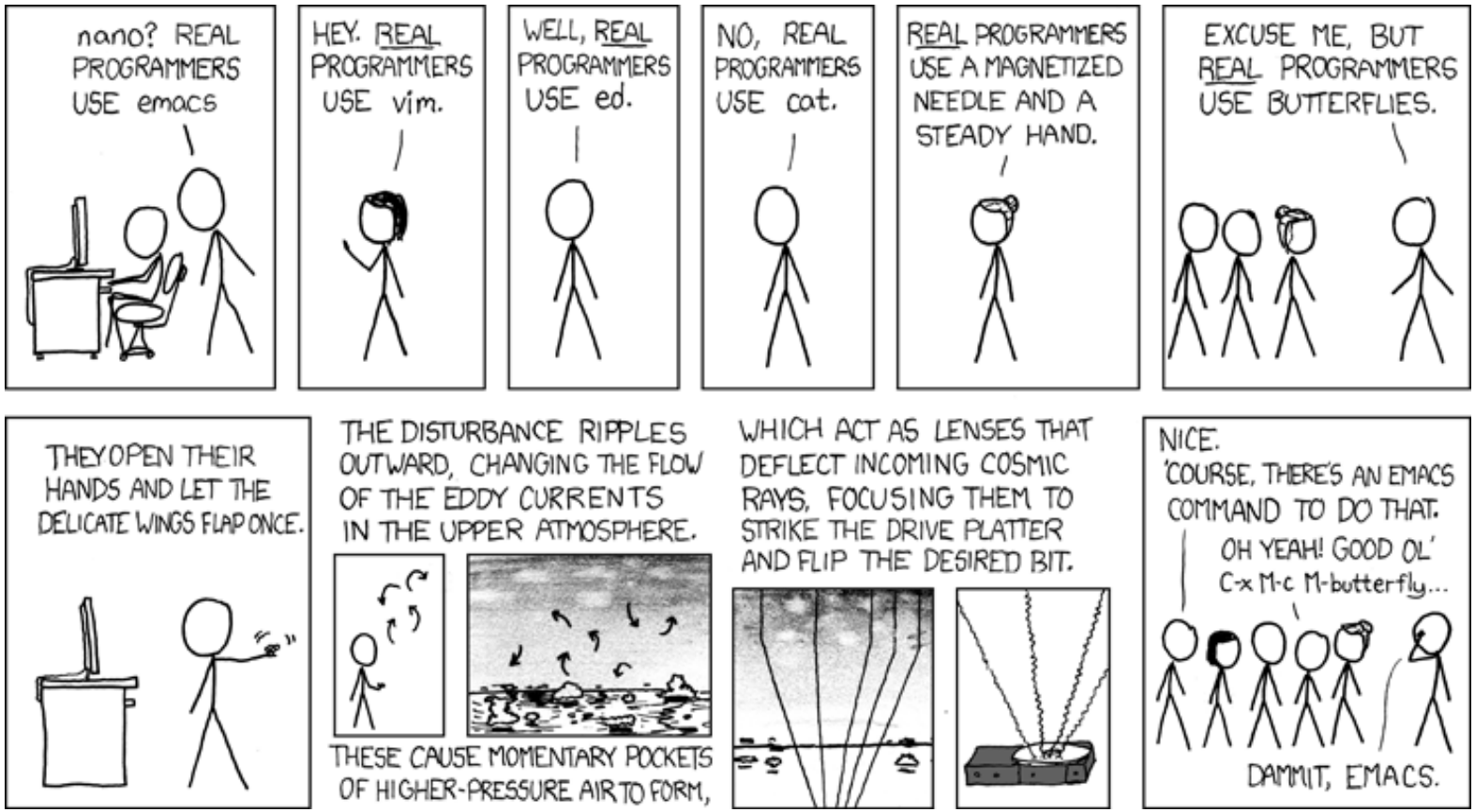
Textdateien betrachten und erstellen und bearbeiten:
Texteditoren, wie z.B.

- Geany
- gedit
- Notepad (Windows)
- emacs (etwas speziell)
- vim (sehr speziell)

# Dateitypen

## Texteditoren





# Zusammenfassung

## Die grundlegendsten Shell-Befehle



| | |
|---|---|
| `pwd` | Anzeigen des aktuellen Verzeichnispfads |
| `ls` | Anzeigen der Dateien in einem Verzeichnis |
| `cd` | Wechseln in anderes Verzeichnis |
| `cp` | Kopieren von Dateien |
| `mv` | Bewegen von Dateien |
| `mkdir` | Erzeugen eines (Unter-)Verzeichnisses |
| `rm` | Löschen von Datei(en)/Verzeichniss(en) |
| `more` | Anzeigen von Dateiinhalten |
| `hexdump` | Binärdateien $\rightarrow$ hexadezimal |

| | |
|---|---|
| `.` | Aktuelles Verzeichnis |
| `..` | Das Verzeichnis darüber |
| `~` | Mein Home-Verzeichnis |

Wildcards:

| | |
|---|---|
| `*` | Ersetzt beliebig viele Zeichen |
| `?` | Ersetzt genau ein Zeichen |
| `[xyz]` | Ersetzt genau ein Zeichen aus x,y,z |

# Ausblick

Nächste Woche machen wir...



## Dateitypen

- Ein- und Ausgabeumleitung
- Ein- und Ausgabeverkettung
- ...

# Ende der heutigen Vorlesung



**Vielen Dank fürs Zusehen!**

**Bis nächste Woche!**