Vorkurs Linux-Grundlagen

Vorkurs Linux / Informatik
Tag 2

Heute:

- Textdateien
- Programme verketten
- Wie funktioniert die Uni

Wiederholung

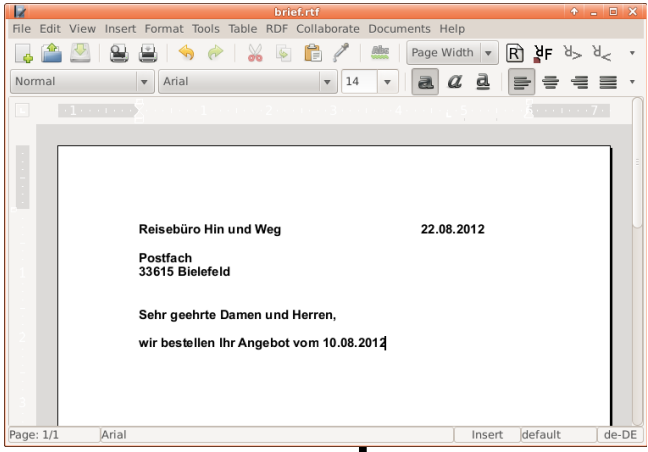
3 Dinge zum Arbeiten mit dem Computer:

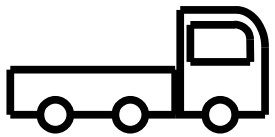
Computer + Programm + Datei =

Computer

AbiWord

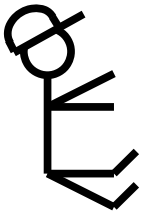
brief.rtf





Auto

+

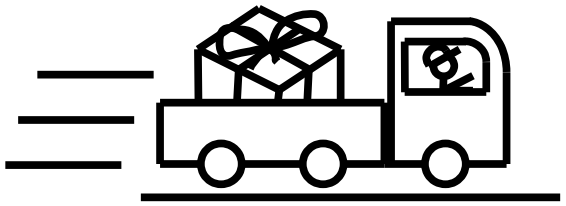


Fahrer

+

Paket

=



Wiederholung

a) Programm aufrufen

```
> abiword
```

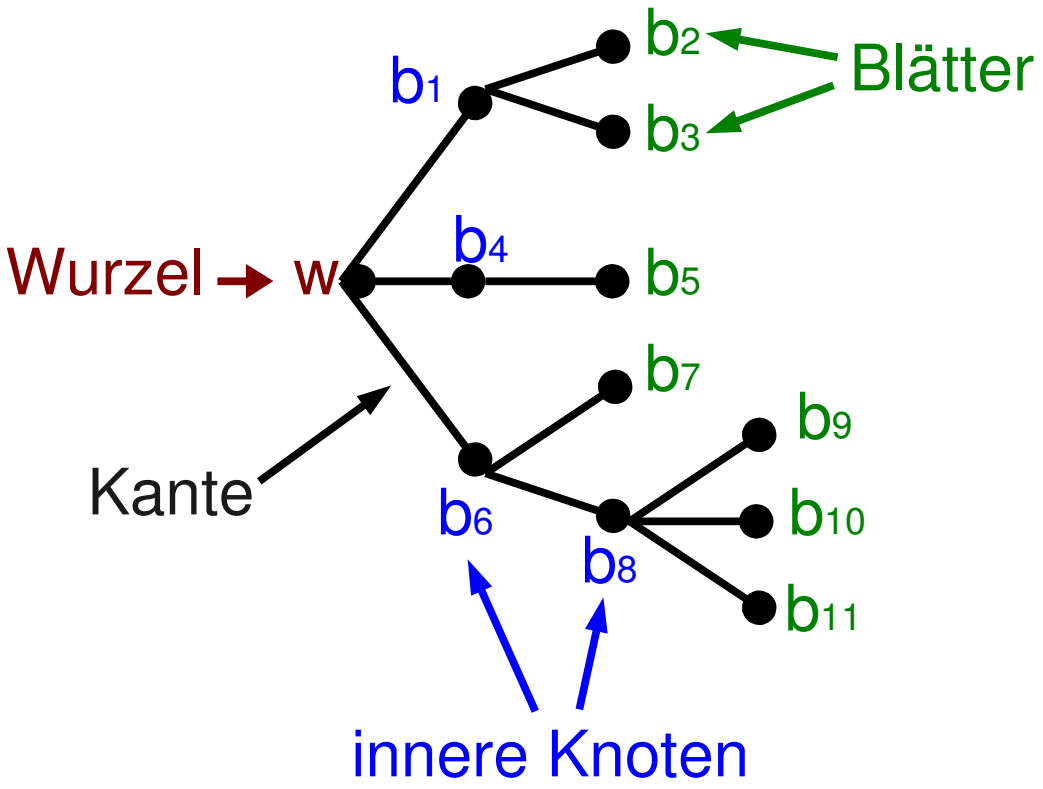
b) Programm mit Datei aufrufen

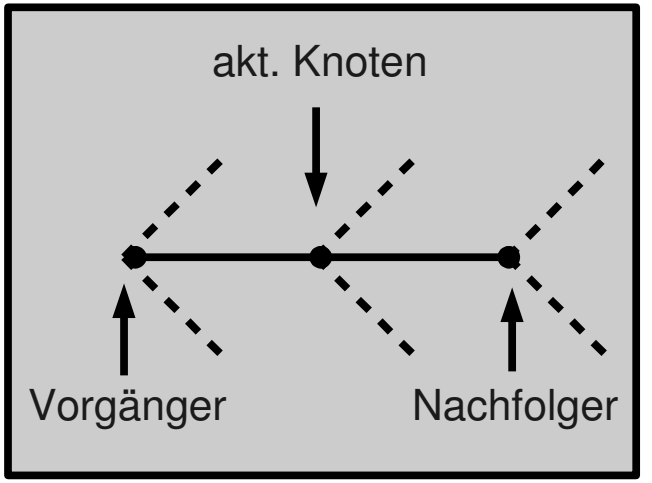
```
> abiword brief.rtf
> idisplay bild.jpg
```

c) Keine Analogie zum Anklicken einer Datei**!**

```
> brief.rtf
bash: brief.rtf:
```
(Fehlermeldung)

Wiederholung





Wiederholung

Arbeiten mit dem Dateisystem

pwd
ls
cd
cp
mv
rm
rmdir

Was machen wir heute?

- Dateitypen: Textdateien, Dokumente, Binärdateien
- Eingabe und Ausgabe von Programmen umleiten
- Ein- und Ausgaben von Programmen verketten
- Werkzeuge zur Bearbeitung von Textdateien
- Einen Exkurs über den Aufbau der Uni

Eine Datei ist eine Folge von Bytes

Dezimal

...	84	101	120	116	...

Hexadezimal (Basis 16)

...	54h	65h	78h	74h	...

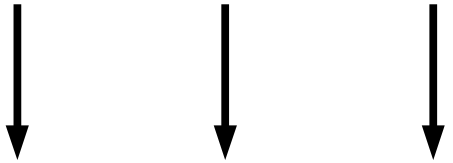
Eine Datei ist eine Folge von Bytes

Dezimal

...	84	101	120	116	...

Hexadezimal (Basis 16)

...	54h	65h	78h	74h	...



willkürliche (!) Abbildung von Bytes auf Buchstaben, Zeichen

ASCII

...	T	e	x	t	...

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)

Elementare Unterschiede (1)

Textdateien ≠ Dokumente

Dokumente sind keine Textdateien!

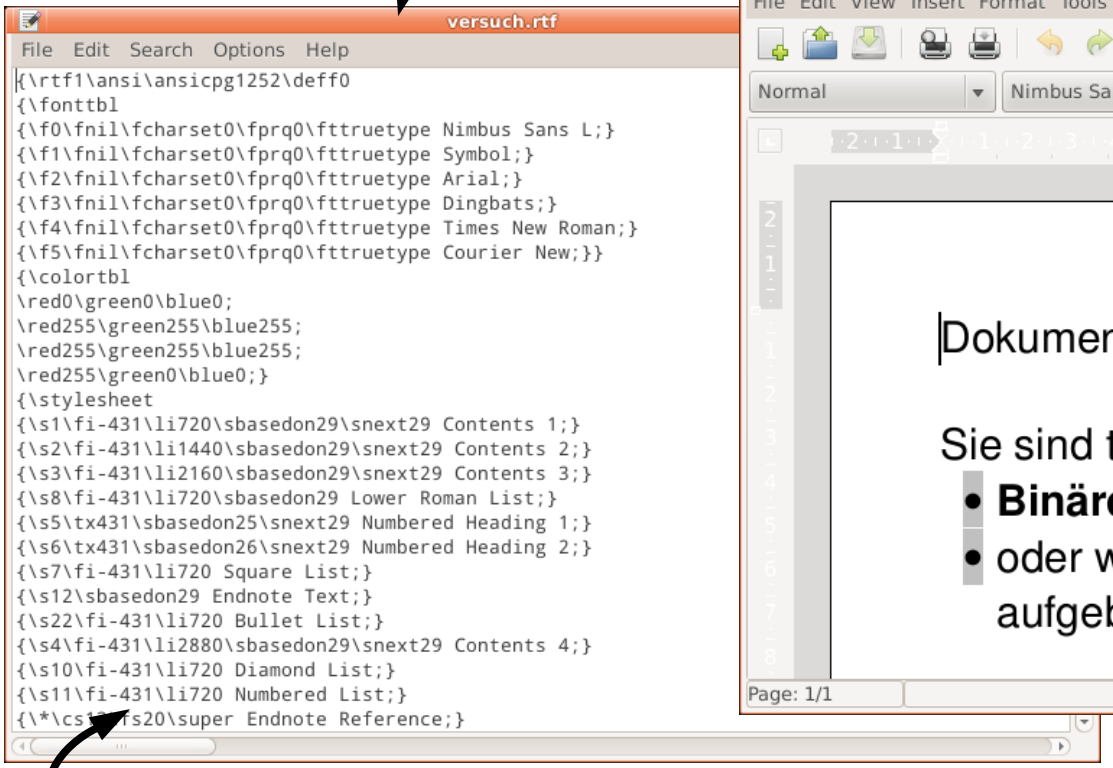
Sie sind
*** Binärdateien oder wie**
*** Programmiersprachen aufgebaut.**

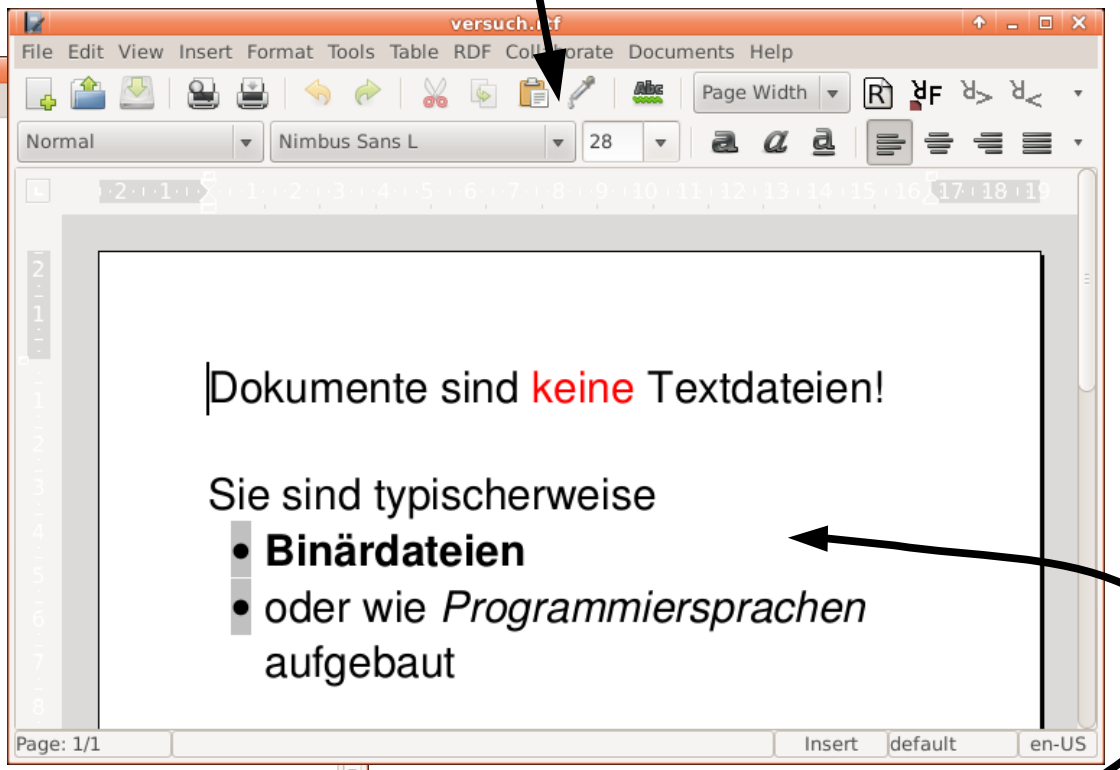
Dokumente sind keine Textdateien!

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

Elementare Unterschiede (2)

Texteditoren ≠ Textverarbeitung





Das steht wirklich in der Datei!

Das seht Ihr in Wordpad!

Beispiele für Textdateien

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

→ **wir arbeiten fast ausschließlich mit Textdateien!**

→ **Finger weg von Word und Co!**

Betrachten von Textdateien

less (Wortspiel; Vorgänger hieß “more”)

```
> less 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 less

Ein- und Ausgabeumleitung

Ein- und Ausgabe sind Text

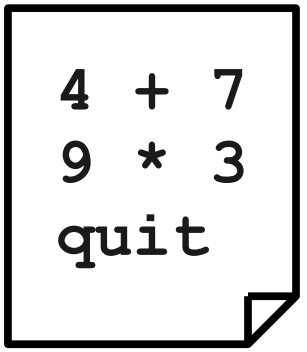
→ Tastatur und Monitor durch Textdateien ersetzbar

Beispiel: interaktive Nutzung

bc (basic calculator) - Kommandozeilen-Taschenrechner

```
> bc
4 + 7
11
9 * 3
27
quit
```

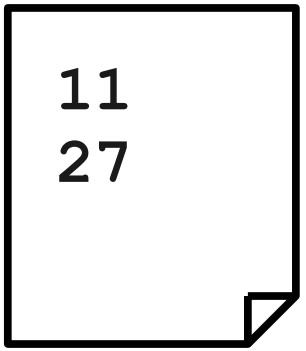
Beispiel: Eingabe-Umleitung




```
> bc < eingabe.txt
11
27
```

Zeichen für Eingabeumleitung!

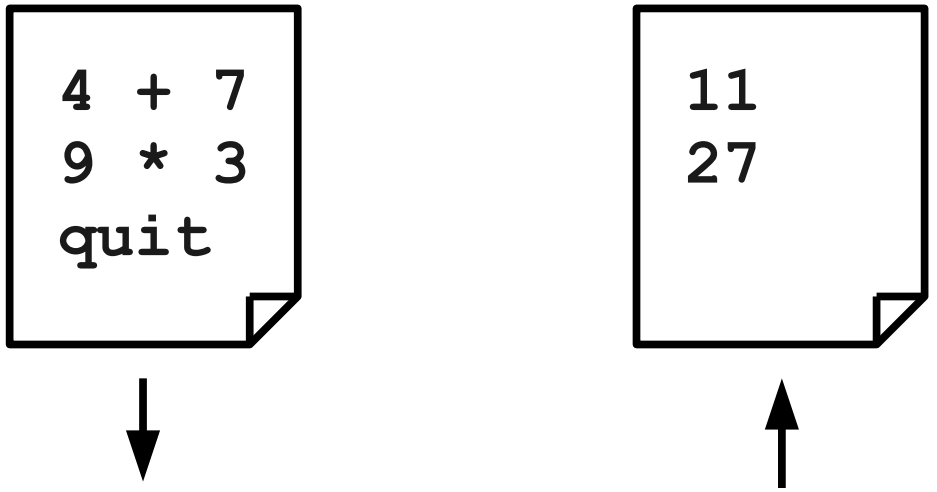
Beispiel: Ausgabe-Umleitung




```
> bc > ausgabe.txt
4 + 7
9 * 3
quit
```

Zeichen für Ausgabeumleitung

Beispiel: beides gleichzeitig




```
> bc < eingabe.txt > ausgabe.txt
```

Textdateien zeilenweise sortieren

sort

```
> sort planeten.txt
```

→ sortiert alphabetisch nach der ersten Spalte

```
> sort -k 2 planeten.txt
```

→ sortiert alphabetisch nach der zweiten Spalte

```
> sort -k 2 -n planeten.txt
```

→ sortiert numerisch nach der zweiten Spalte

Verzeichnis nach Größe sortieren

- `ls -l > zwischen.txt`
- `sort -k 5 -n zwischen.txt > sort.txt`
- `less sort.txt`
- `rm zwischen.txt sort.txt`

→ das Hantieren mit temporären Dateien ist lästig!

Idee: Ein-/Ausgabeweiterleitung

Eingabe

Programm1

Programm2

...

Programmn

Ausgabe

Beispiel: Ein-/Ausgabeweiterleitung

„Pipe“ - Symbol | verbindet die Programme:

```
> ls -l | sort -k 5 -n | less
```

· Ausgabe des links von | stehenden Programms

· wird Eingabe des rechts von | stehenden Progr.

→ deutlich effizienter als Zwischenspeichern!

Textdateien zusammenfügen

cat : concatenate files

```
> cat eins.txt zwei.txt drei.txt
```

gibt den Inhalt der Dateien nacheinander aus.

```
> cat eins.txt zwei.txt drei.txt >sammlung.txt
```

Ergebnis in neuer Datei speichern

```
> cat eins.txt
```

Nützlicher Spezialfall: Eine kurze Datei anschauen

Dateien zeilenweise vergleichen (1)

diff : show differences between files

```
> diff links.txt rechts.txt
```

Entziffern der Ausgabe von diff:

ncm: Die nachfolgenden Zeilen wurden verändert.
< Zeilen von links.txt; > Zeilen von rechts.txt

```
6c8
< einen Gegenstand zu kaufen, wird er,
---
> einen Gegenstand zu ersteigern, wird
```

Dateien zeilenweise vergleichen (2)

*n*a*m:* in der Datei *rechts.txt* hinzugefügte Zeilen (added)

4a5,6
> Mit dieser Datei kann man diff ausprobieren.
> ← (hinzugefügte Leerzeile)

*n*d*m:* in der Datei *rechts.txt* gelöschte Zeilen (deleted)

11,12d12
< ← (mitgelöschte Leerzeile)
< -- Originalversion

Anfang einer Datei ansehen

head : show head of file

```
> head -3 eins.txt
```

zeigt die ersten 3 Zeilen einer Datei.

Ende einer Datei ansehen

tail : show tail of file

```
> tail -4 zwei.txt
```

zeigt die letzten 4 Zeilen einer Datei

```
> tail -n +7 eins.txt
```

zeigt alle Zeilen ab der 7ten Zeile
(bzw. unterdrückt die Zeilen 1 bis 6)

Komplexeres Beispiel (1)

Planeten-Tabelle mit Überschrift sortieren

```
> sort planeten2.txt
```

→ klappt nicht wegen der Überschrift

Ansatz: Überschrift mit tail abschneiden

```
> tail -n +3 planeten2.txt | sort
```

Komplexeres Beispiel (2)

Überschrift erhält man mit head:

```
> head -2 planeten2.txt
```

Alles zusammenfügen:

```
> head -2 planeten2.txt > teil1.txt
> tail -n +3 planeten2.txt | sort > teil2.txt
> cat teil1.txt teil2.txt > neu.txt
> rm teil1.txt teil2.txt
```

Komplexeres Beispiel (3)

Es geht auch ohne Zwischendateien:

```
> head -2 planeten2.txt; tail -n +3 planeten2.txt|sort
```

Semikolon trennt Aufrufe

→ man kann mehr als ein Programm pro Zeile ausführen

Ausführung von links nach rechts

Ausgaben werden aneinandergehängt

Texte in Dateien suchen

grep : global regular expression print

```
> grep datei *.txt
drei.txt:3 dritte Textdatei
eins.txt:1 erste Textdatei
links.txt:* noch eine Textdatei
...
```

Durchsucht alle Dateien mit der Endung *.txt, ob sie den Text “datei” enthalten.

```
> grep -i datei *.txt
```

Ausgaben mit grep filtern

Filtern von Programmausgaben mit grep:

```
> ls -la | grep 2012
```

→ alle Dateien mit Datum 2012 zeigen

Texteditoren

emacs – Texteditor

- sehr mächtiger Texteditor

  → gut für Programmierung, Shellskripte

  → erweiterbar und programmierbar

- mit Maus aber auch komplett über Tastatur bedienbar

  → schrittweiser Umstieg auf Tastatur (schneller, mächtiger)

Aufruf von Emacs

Normale Kommandozeilen-Programme

- benötigen Kommandozeilen-Fenster für Ein/Ausgabe
- Beispiel: **`bc`** blockiert das Fenster, solange es läuft

Emacs hat eigenes Fenster

- „Abkoppeln“ von der Kommandozeile: **`> emacs &`**

→ Kommandozeile weiter verwenden, während emacs läuft

- sinnvoll für Programme, die eigenes Fenster öffnen

Weitere Texteditoren

- vim/gvim

  sehr mächtig, modusbasiert
- gedit

  simpler, weniger Abkürzungstasten
- nano

  konsolenbasiert, simpel, eher wie Notepad
- joe

  konsolenbasiert, simpel + Abkürzungstast.

Exkurs: Wie funktioniert die Uni?

Wechseln zum anderen Foliensatz

Ende des heutigen Vortrags

Danke fürs Zuhören!

Bis morgen