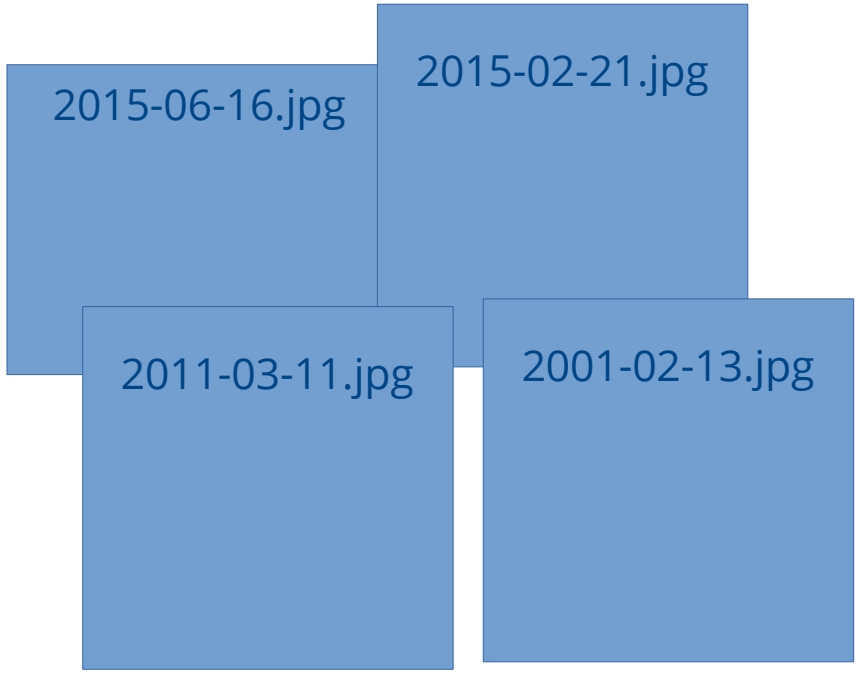
Introduction to Linux/bash

Carl Herrmann
Cancer Regulatory Genomics
B080 - DKFZ

mit Hilfe von D. Puthier
TAGC Marseille

Imagine ...

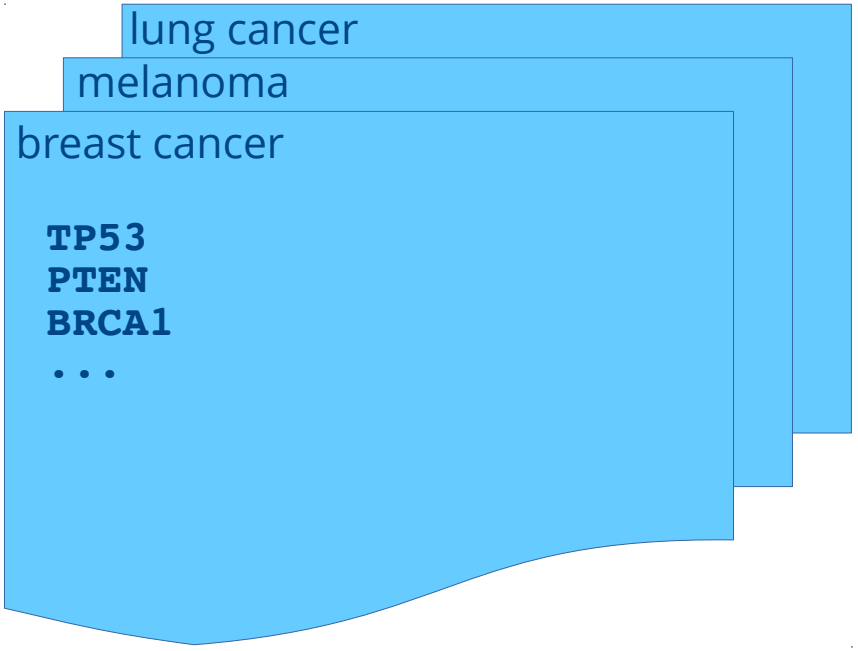


- ich habe 10.000 Fotos in jpeg Format
- Wie kann ich sie automatisch entsprechend des Filenames in Unterordner organisieren ?

Imagine ...

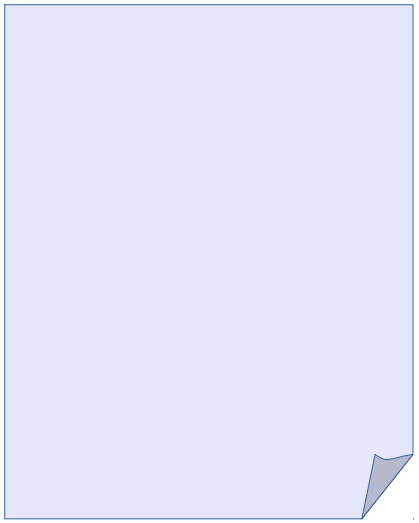
- ich habe eine Liste von Genen, die in verschiedenen Tumorarten mutiert sind

- Wie finde ich, welche gene ...
  - nur in einer Tumorart mutiert sind ?
  - in mindestens 2 ?
  - überhaupt nicht mutiert sind ?



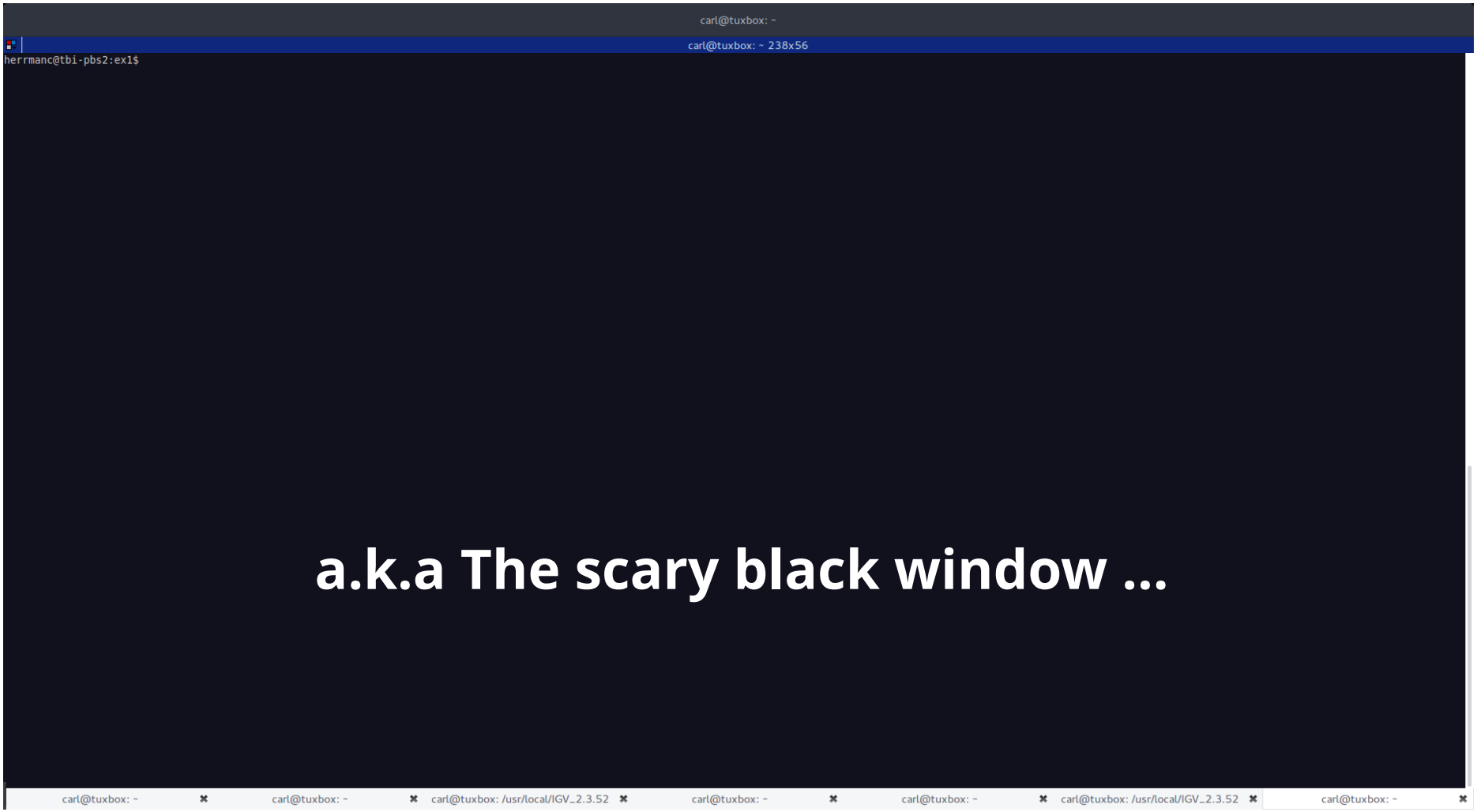
Imagine ...

- Ich habe eine 4.7 Gigabyte grosse Textdatei
- Wie schaue ich mir die ersten 10 Zeilen an ?



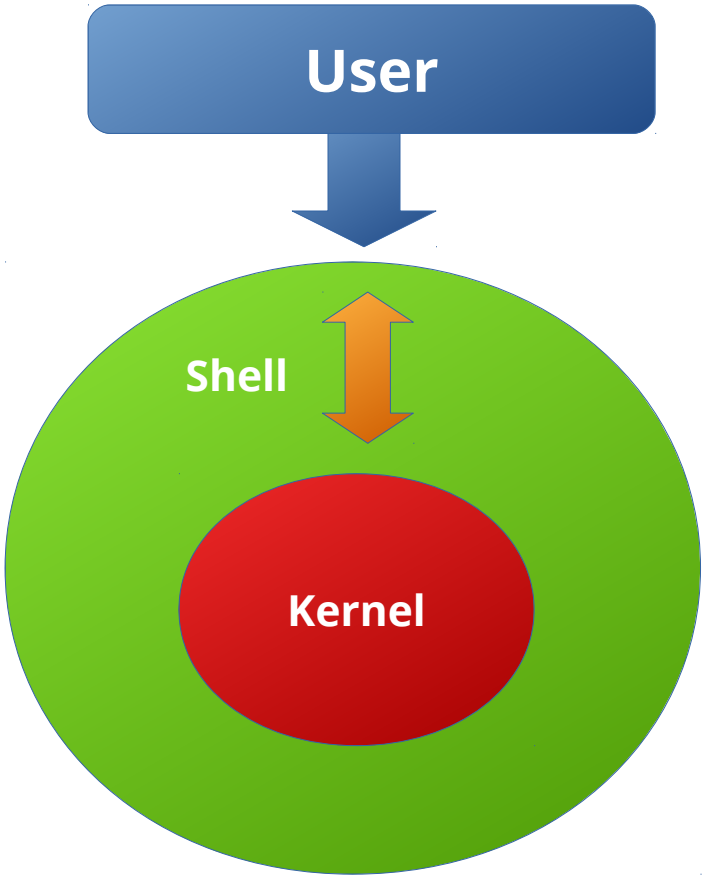
```
SQ     SN:gi|254160123|ref|NC_012967.1|       LN:4629812
@PG    ID:bwa  PN:bwa  VN:0.5.9-r16
SRR098038.1     0       gi|254160123|ref|NC_012967.1|   2834490 37      36M     *       0       0       GTTTATTTTTTCTTGTGAAGGGATCGATAGCAGCGG    CCCCCCCCCCCCCCC<C-CC*-6CCC<CCC6;;;?65    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
SRR098038.2     16      gi|254160123|ref|NC_012967.1|   2871817 37      36M     *       0       0       GATGCCCTTGCTGAAAGTGGTATTGCGGCCAATGAC    =?A??A5CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
SRR098038.3     0       gi|254160123|ref|NC_012967.1|   2208208 37      36M     *       0       0       AATCGCCGGAAGTACGCGAAAGGCGAAGATAAAACC    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC??A??>    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
SRR098038.4     4       *       0       0       *       *       0       0       GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGGAT    CCCCCCCCCCCCCCCCCC<CCCCCCC>C61??%)=-
SRR098038.5     0       gi|254160123|ref|NC_012967.1|   4154304 37      36M     *       0       0       ATAGCGCCTGCAATACACGGGCGGTTGTACTTTTCC    CCCCCCCCCCCCCCCCCCCCCCCCC;C<<95???A-8    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
SRR098038.6     0       gi|254160123|ref|NC_012967.1|   1962282 37      36M     *       0       0       AGGCGCTCCTCGCCATAACTACGCGGGTGCCAGCCG    CCCCCCCCCCCCCCCCCCCCCCCCCC#C65???=?5    XT:A:U  NM:i:1  X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0
SRR098038.7     0       gi|254160123|ref|NC_012967.1|   2322479 37      36M     *       0       0       GGATTCCGTTTGAATTTGCTGAAGTCAGGCTGACGT    CCCCCCCCCCCCCCCCCCCC<CCACCC6A@A;?;=?    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
SRR098038.8     4       *       0       0       *       *       0       0       GATCGGAAGGCTCGTATGCCGTCTTCTGCTTGGATC    CCCCCCCCCCCCCCCCCCCCCCCCCCCCAAC?"6=5)
SRR098038.9     0       gi|254160123|ref|NC_012967.1|   1216418 37      36M     *       0       0       GATTCAGGAGGGCCAAACCGAAGAACAGCGACAGCT    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCA??=;=    XT:A:U  NM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
```

Mit der Kommandozeile und bash



Wie kann man damit kommunizieren ??

- **Linux** ist ein freies, offenes Betriebssystem, das auf **Unix** basiert.
- Mac OS basiert mitlerweile auch auf einem Linux **Kernel**
- Windows nicht ...
- In Linux spricht man vom
  - **Kernel** = Herzstück des Betriebssystems
  - **Shell** = Schicht zwischen dem Benutzer und dem Kernel
- Es gibt verschiedene Shells (=Dialekte) : tsh, csh, ksh
- Wir benutzen **bash** (= bourne again shell)



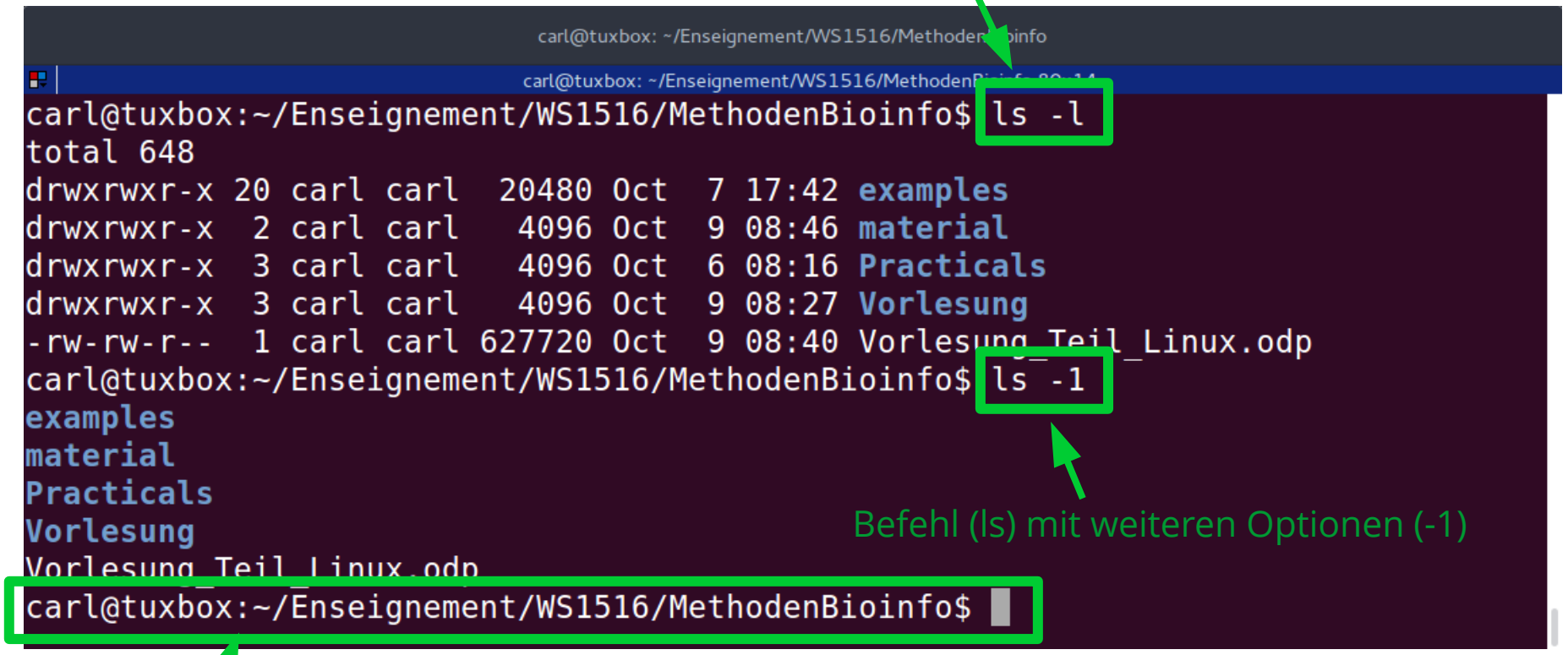
Wie kann man damit kommunizieren ??

- Bash ist eine eigene Sprache
- Besteht aus **Kommandos**, die Abkürzungen aus dem Englishen sind

```
ls  →   list    = zeige den Inhalt des aktuellen Verzeichnisses
pwd →   print working directory = zeige das aktuelle
Verzeichnis
uvm ...
```

Die Kommandozeile

Befehl (ls) mit weiteren Optionen (-l)

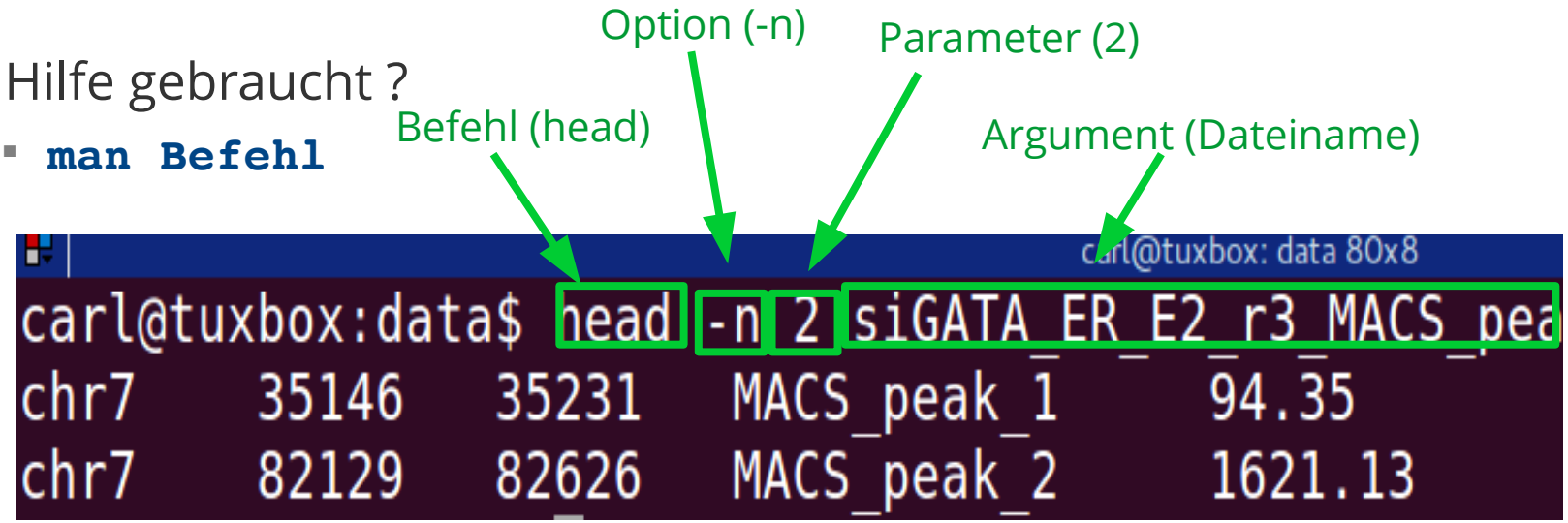


Befehl (ls) mit weiteren Optionen (-1)

"prompt" : wartet auf Eingabe ; gib hier Benutzername (carl), Name der Machine (tuxbox), und aktuelles Verzeichnis (~/Enseignement/WS1516/MethodenBioinfo)

Befehle

- zurück zum Vokabellernen ... :-(
- es gibt **~ 15 Grundbefehle**, mit denen man sich verständigen kann
- Struktur :
  - Grundbefehl
  - Optionen, mit oder ohne zusätzliche Parameter
- Hilfe gebraucht ?
  - **`man Befehl`**

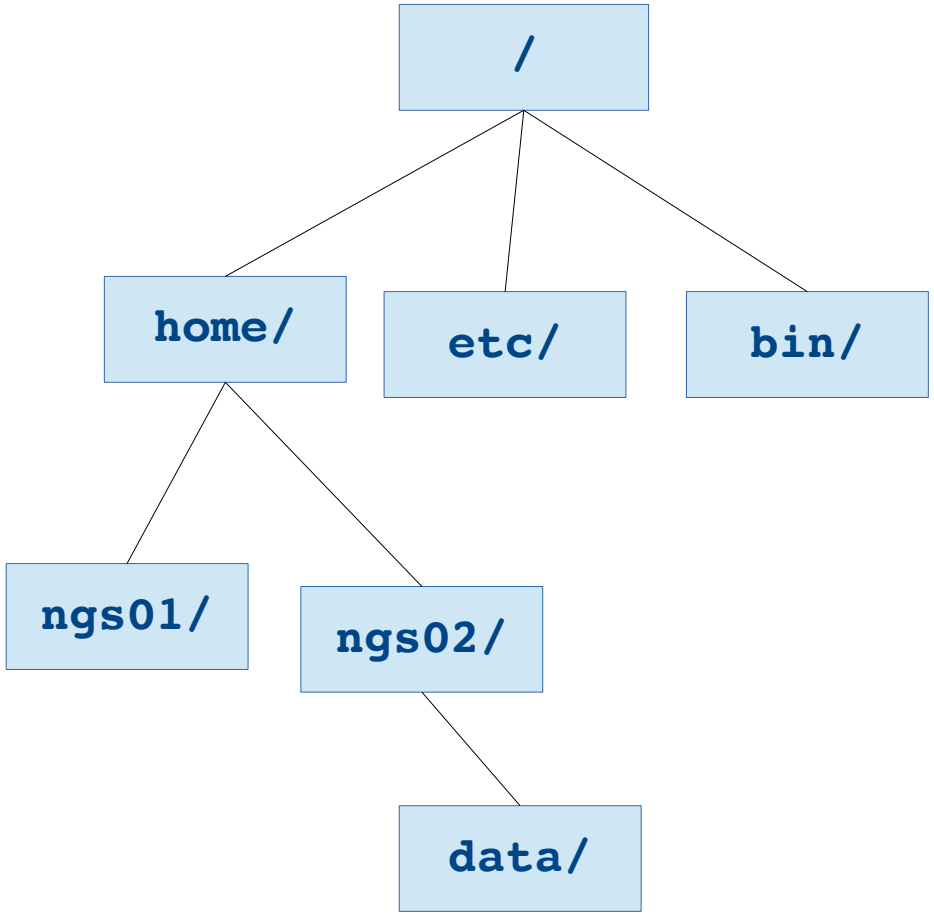


Optionen werden mit einem “-” Zeichen angegeben

Sich bewegen …

Struktur der Verzeichnisse

- Verschachtelte Struktur
- jeder Benutzer hat ein **Heimatverzeichnis**, meistens /home/<user>
- einige Besondere Verzeichnisse:
  - “root directory” : /
  - das aktuelle Verzeichnis : ./
    - aktuelles Verzeichnis → ngs01/
    - ./ → ngs01/
  - das Verzeichnis über dem aktuellen : ../
    - aktuelles Verzeichniss → ngs01/
    - ../ → home/
  - Heimatverzeichnis : ~/

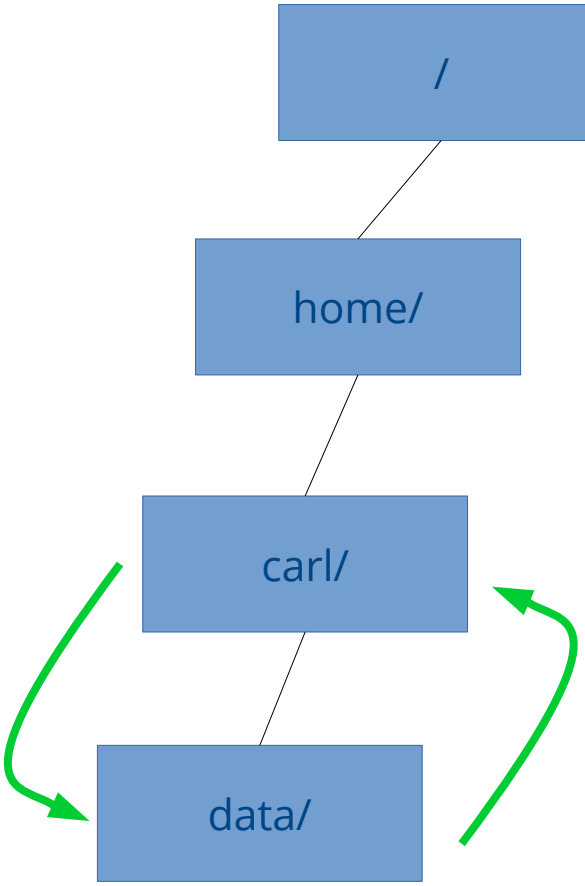


Sich bewegen ...

wo bin ich ??

cd → change directory

pwd → print working directory

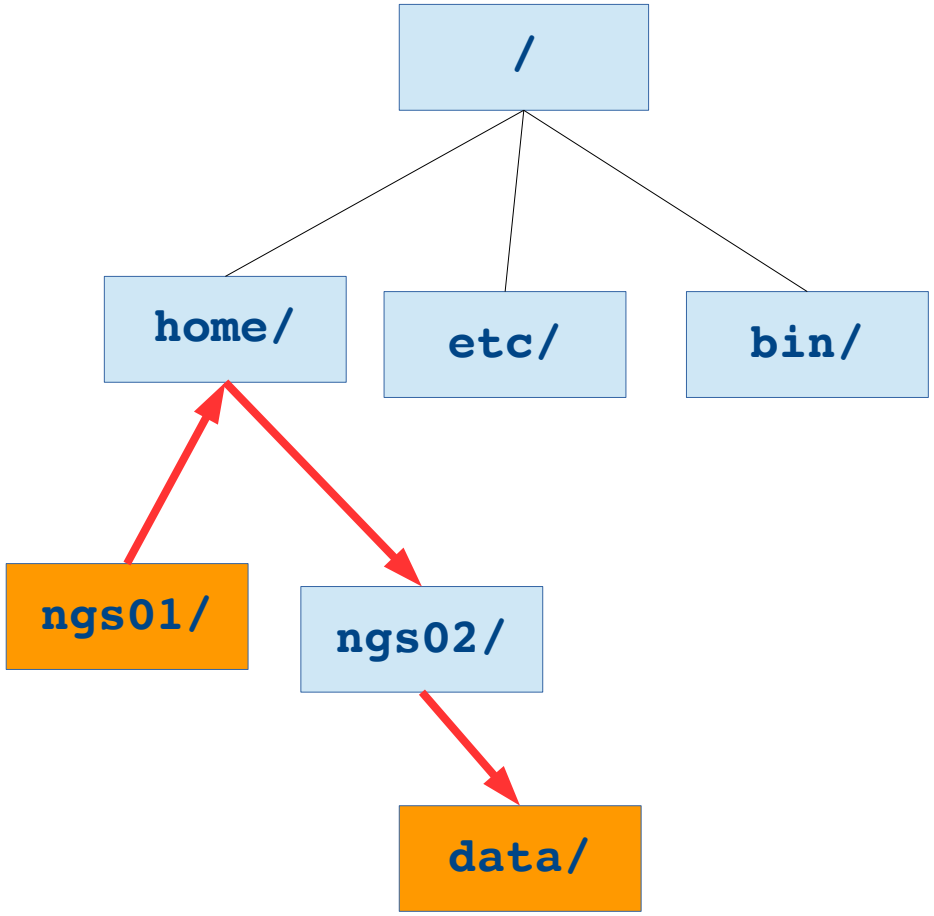



```
carl@tuxbox: ~
carl@tuxbox: ~ 46x17
carl@tuxbox:~$ # wo bin ich gerade
carl@tuxbox:~$ pwd
/home/carl
carl@tuxbox:~$ # in ein Unterverzeichnis wechs
eln
carl@tuxbox:~$ cd data/
carl@tuxbox:data$ # wo bin ich gelandet ?
carl@tuxbox:data$ pwd
/home/carl/data
carl@tuxbox:data$ # jetzt wieder rauf
carl@tuxbox:data$ cd ../
carl@tuxbox:~$ pwd
/home/carl
carl@tuxbox:~$
```

Pfade : wie findet man sich zurecht ?

- mein aktuelles Verzeichnis :
  `/home/ngs01/`

- Das Verzeichnis data/ kann ich **relativ zu meinem aktuellen Verzeichnis** angeben :
  - einmal nach oben ...
  - dann in ngs02 ...
  - dann in data ...




```
cd ../ngs02/data/
```

Pfade : wie findet man sich zurecht ?

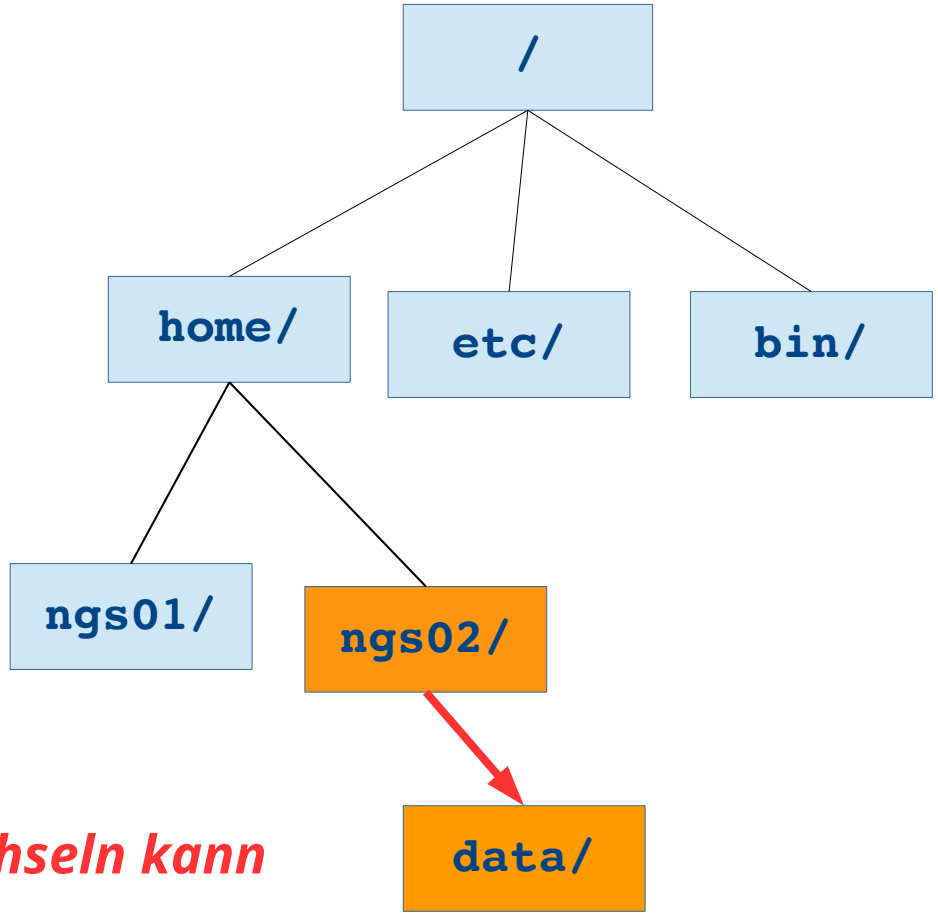
- mein aktuelles Verzeichnis :
  `/home/ngs02/`

- ich möchte in data/ wechseln: 2 Möglichkeiten

```
cd ./data/
```

oder

```
cd data/
```



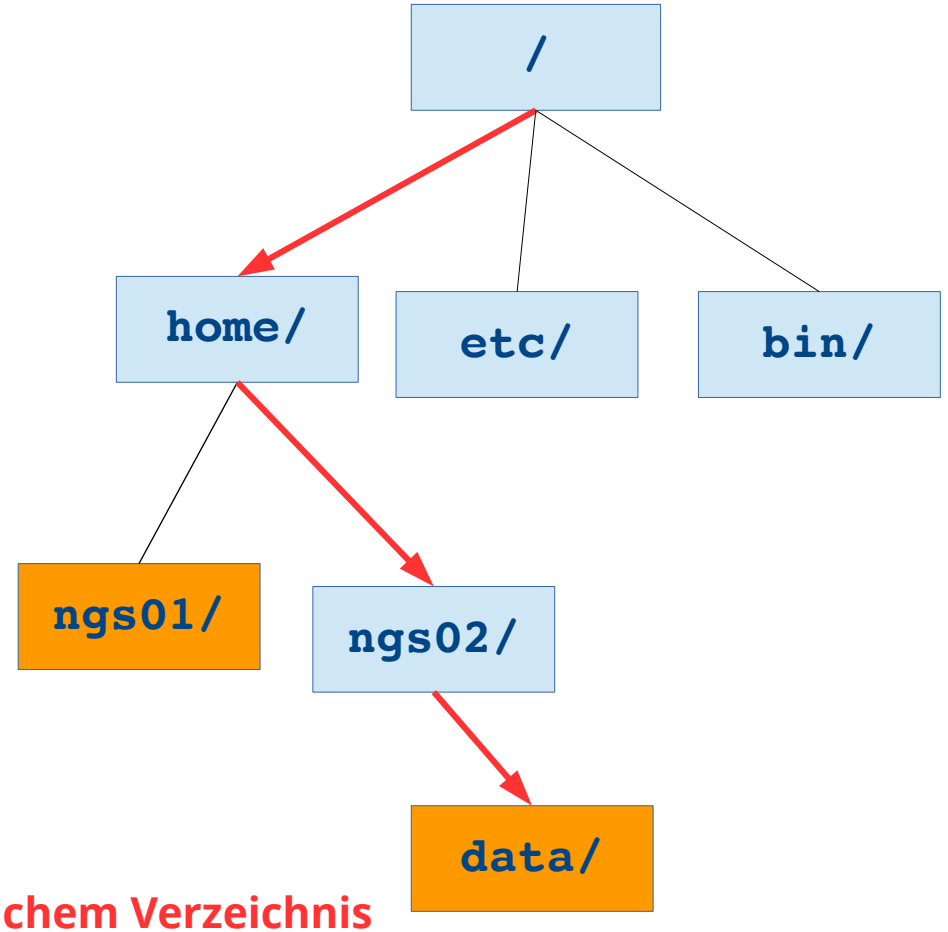
Um in ein Unterverzeichnis zu wechseln kann man das ./ weglassen !

Pfade : wie findet man sich zurecht ?

- mein aktuelles Verzeichnis :
  `/home/ngs01/`

- Das Verzeichnis data/ kann ich **absolut ab dem root Verzeichnis** angeben :
  - von root ausgehend ...
  - dann in home/ ...
  - dann in ngs02/
  - dann in data/

```
cd /home/ngs02/data/
```

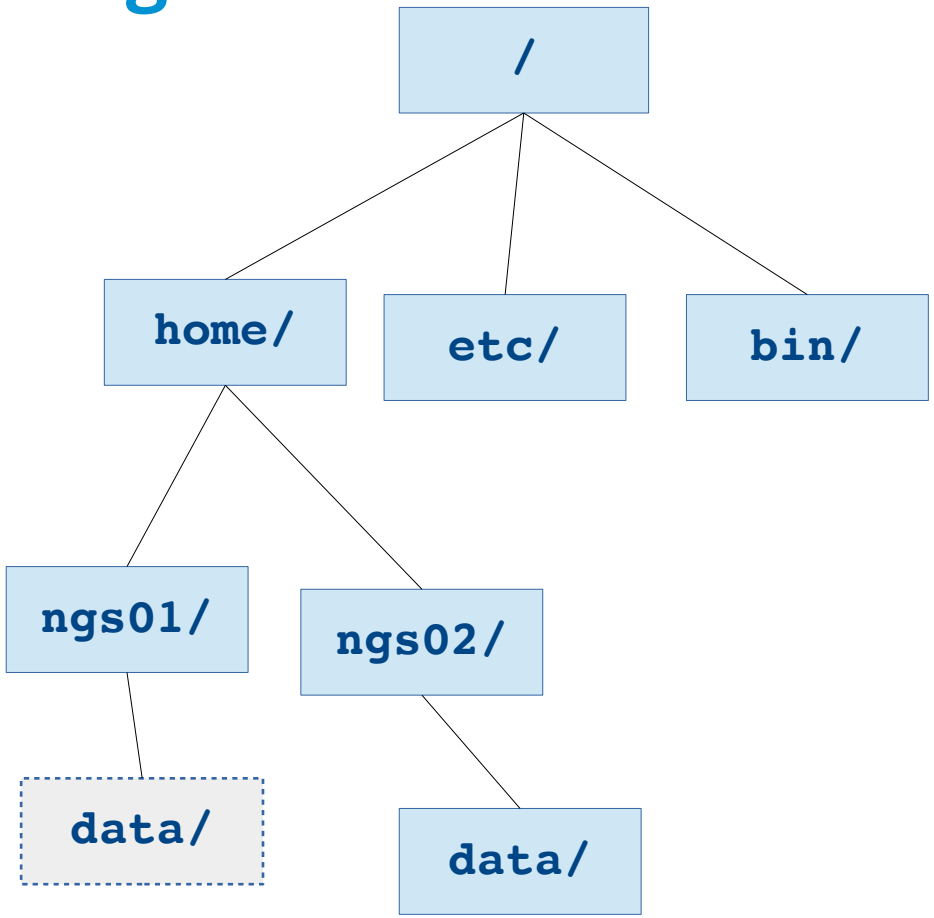


Absolute Pfade funktionnieren, egal in welchem Verzeichnis ich mich gerade befinde !

Ein neues Verzeichnis anlegen

`mkdir → make directory`

- Unterverzeichnis **`data/`** anlegen:




```
mkdir data/
```

Relativ : Funktionniert nur, wenn ich in /home/ngs01/ bin !

```
mkdir /home/ngs01/data/
```

Absolut : Funktionniert immer !

Faustregel

- fängt der Pfad mit **“/”** oder **“~”** an
  → **absoluter Pfad** (startet also vom “Root”-Verzeichnis)
  - `cd /home/ngs01/data`
  - `cd ~/data`

- fängt der Pfad mit **“.”** oder **einem Namen**
  → **relativer Pfad** (startet vom aktuellen Verzeichnis)
  - `cd ../data/annotations`
  - `cd data`
  - `cd ./data`

Wichtige Regeln !

- Dateinamen und Verzeichnisse **ohne Leerzeichen**
  - alphanumerische Zeichen
  - statt Leerzeichen das “_” benutzen !

```
meine_erste_Datei.txt
```

- Dateinamen mit **Endungen** versehen:
  - Textdateien : .txt
  - Bash Skripte : .sh
  - Python Skripte : .py
  - etc ...

- Dadurch erkennt man sofort den Typ der Datei.

Den Inhalt eines Verzeichnisses anschauen

`ls → list`

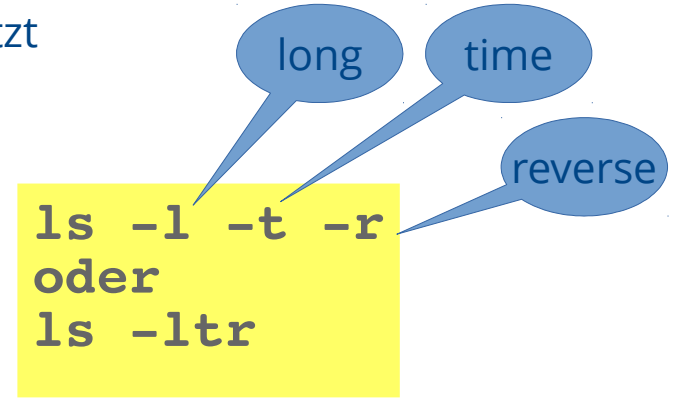
Einfaches auflisten

```
ls
```

Auflisten mit zusätzlichen Informationen

```
ls -l
```

Auflisten mit zusätzlichen Informationen, die “jüngsten” Dateien zuletzt




```
carl@tuxbox: MethodenBioinfo
carl@tuxbox: MethodenBioinfo 78x17
carl@tuxbox:MethodenBioinfo$ ls
examples  material  Practicals  Vorlesung  Vorlesung_Teil_Linux.odp
carl@tuxbox:MethodenBioinfo$ ls -l
total 940
drwxrwxr-x 20 carl carl  20480 Oct  7 17:42 examples
drwxrwxr-x  2 carl carl   4096 Oct  9 08:46 material
drwxrwxr-x  3 carl carl   4096 Oct  6 08:16 Practicals
drwxrwxr-x  3 carl carl   4096 Oct  9 08:27 Vorlesung
-rw-rw-r--  1 carl carl 928885 Oct  9 10:01 Vorlesung_Teil_Linux.od
carl@tuxbox:MethodenBioinfo$ ls -ltr
total 940
drwxrwxr-x  3 carl carl   4096 Oct  6 08:16 Practicals
drwxrwxr-x 20 carl carl  20480 Oct  7 17:42 examples
drwxrwxr-x  3 carl carl   4096 Oct  9 08:27 Vorlesung
drwxrwxr-x  2 carl carl   4096 Oct  9 08:46 material
-rw-rw-r--  1 carl carl 928885 Oct  9 10:01 Vorlesung_Teil_Linux.od
carl@tuxbox:MethodenBioinfo$
```

Dateien kopieren / umbenennen

Kopieren / umbenennen

- Befehl cp (=copy)
  - **`cp <Datei> <Verzeichnis>`**
    → kopiert die Datei in das Verzeichnis
  - **`cp <Datei> <Verzeichnis>/<neuer Name>`**
    → kopiert die Datei in eine neue Datei mit neuem Namen

- Befehl mv (=move)
  - **`mv <Datei> <Verzeichnis>`**
    → verschiebt die Datei in das Verzeichnis
  - **`mv <Datei> <Verzeichnis>/<neuer Name>`**
    → verschiebt die Datei in eine neue Datei mit neuem Namen

Mit Dateien arbeiten

Dateientypen

- **Binäre Dateien**
  - brauchen eine besondere Software, um bearbeitet zu werden
    - Bilder → jpeg, png
    - Ton → mp3, ogg, wav
    - Sequenzalignments → bam

- **Text Dateien** (oder ASCII-Dateien)
  - bestehen aus ASCII-Zeichen
  - können mit jedem Texteditor aufgerufen werden

- Bash verfügt über zahlreiche Befehle, um Textdateien zu bearbeiten

Inhalt einer Datei anschauen

- Befehle **`less`** (oder **`more`** aber **`less`** ist besser...) ermöglichen Einsicht in die Datei
  - runter/rauf mit Pfeil nach unten/oben
  - rausgehen mit q (=quit)
  - etwas suchen mit : `/text`

- Befehlen **`head`** oder **`tail`** erlauben die ersten/letzten Zeilen einer Datei anzuschauen
  - head file.txt → erste 6 Zeilen
  - head -n 10 file.txt → erste 10 Zeilen

- Befehl **`cat`** um den gesamten Inhalt der Datei anzuzeigen (Achtung bei grossen Dateien!)
  - <Strg> + C um die Ausgabe zu stoppen

Zeilen / Wörter / Zeichen zählen

- Befehl **`wc`** (=word count)
  - gibt Anzahl der Zeilen / Wörter / Zeichen
  - **`wc -l`** → nur Anzahl von Zeilen

```
carl@tuxbox:material$ wc hg19_exons.bed
  484127  2904762 18963872 hg19_exons.bed
carl@tuxbox:material$
carl@tuxbox:material$ wc -l hg19_exons.bed
484127 hg19_exons.bed
carl@tuxbox:material$
carl@tuxbox:material$
carl@tuxbox:material$ head hg19_exons.bed
chr4    2271323 2272583 NM_001172656    0       -
chr4    2273037 2273141 NM_001172656    0       -
chr4    2273401 2273506 NM_001172656    0       -
chr4    2274899 2275016 NM_001172656    0       -
chr4    2275788 2275943 NM_001172656    0       -
chr4    2306015 2307263 NM_001172656    0       -
chr4    2321896 2321998 NM_001172656    0       -
chr4    2339133 2339223 NM_001172656    0       -
chr4    2341179 2341382 NM_001172656    0       -
chr4    2343204 2343342 NM_001172656    0       -
carl@tuxbox:material$
carl@tuxbox:material$
carl@tuxbox:material$ tail hg19_exons.bed
chr20   2442280 2442585 NM_003091       0       -
chr20   2443281 2443407 NM_003091       0       -
chr20   2443734 2443873 NM_003091       0       -
chr20   2444392 2444545 NM_003091       0       -
chr20   2446353 2446465 NM_003091       0       -
chr20   2448252 2448404 NM_003091       0       -
chr20   2451333 2451499 NM_003091       0       -
chr6    41873090        41875025        NR_131160       0       -
chr6    41884522        41884677        NR_131160       0       -
chr6    41888742        41888885        NR_131160       0       -
carl@tuxbox:material$
```

Spalten ausschneiden

- Befehl **`cut`**
  - **`cut -f <Spaltennummer> <Dateiname>`**

- **Achtung** ! Spalten müssen durch Tabulierungen getrennt werden; wenn nicht, muss der “Separator” mit **`-d`** angegeben werden

```
cut -f1 hg19_exon.bed                  # gibt die erste Spalte
cut -f1,2 hg19_exon.bed                # gibt die beiden ersten
                                       # Spalten
cut -f1-4 hg19_exon.bed                # gibt Spalten 1 bis 4
cut -d “,” -f1 hg19_exon.csv           # gibt erste Spalte wenn
                                       # Spalten durch Komma getrennt
                                       # sind
```

Dateien sortieren

- Befehl **`sort`**
  - **`sort -k <Spaltennummer> <Datei>`**
    → sortieren nach Spalte
  - **`sort -k <Spaltennummer> -r <Datei>`**
    → sortieren nach Spalte in umgekehrter Reihenfolge

- Achtung !
  - **`sort`** → sortiert alphabetisch
  - **`sort -n`** → sortiert numerisch !
  - **`sort -g`** → sortiert numerisch mit dezimalen Zahlen

Das Verhalten des Befehls sort kann je nach Spracheinstellung des Systems verschieden sein !

```
carl@tuxbox:material$ cat test.txt
1
10
12
6
21
32
11
9
18
carl@tuxbox:material$ sort test.txt
1
10
11
12
18
21
32
6
9
carl@tuxbox:material$ sort -n test.txt
1
6
9
10
11
12
18
21
32
carl@tuxbox:material$
```

Eine Zeichenkette suchen

- Befehlt grep
  - **`grep <Zeichenkette> <Datei>`**
  - **`grep -i <Zeichenkette> <Datei>`**
    → -i : unterscheidet NICHT Gross-/Kleinschreibung
  - **`grep -w <Zeichenkette> <Datei>`**
    → -w : sucht nach der GENAUEN Zeichenkette

```
Anna
Annamaria
Anna-Maria
carl@tuxbox:material$ grep -w Anna namen.txt
Anna
Anna-Maria
carl@tuxbox:material$ grep -i Anna namen.txt
Anna
Annamaria
Hanna
Anna-Maria
carl@tuxbox:material$ grep -i -w Anna namen.txt
Anna
Anna-Maria
carl@tuxbox:material$
```

Anna
Annamaria
Hanna
Paul
Anna-Maria

Eine Zeichenkette NICHT suchen

- Befehlt grep
  - **`grep -v <Zeichenkette> <Datei>`**
    → sucht Zeilen, die die Zeichenkette NICHT enthalten

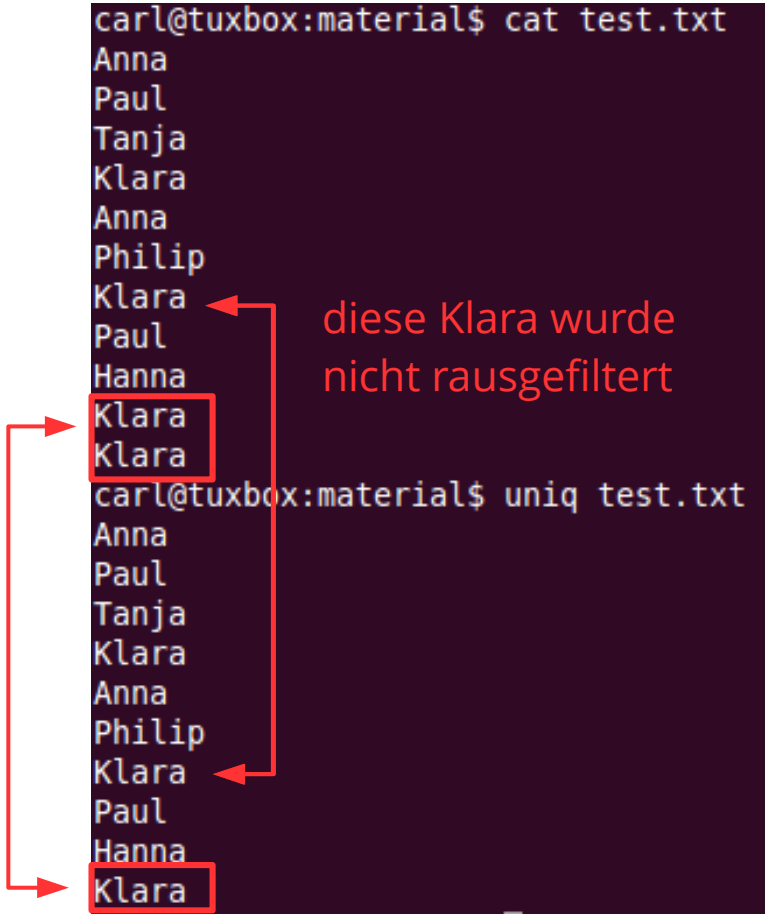
Wie bekomme ich nur die Zeile Paul ??

```
carl@tuxbox:material$ grep -i -v  Anna namen.txt
Paul
carl@tuxbox:material$
```

Anna
Annamaria
Hanna
Paul
Anna-Maria

Doppelt vorhandene Zeilen filtern

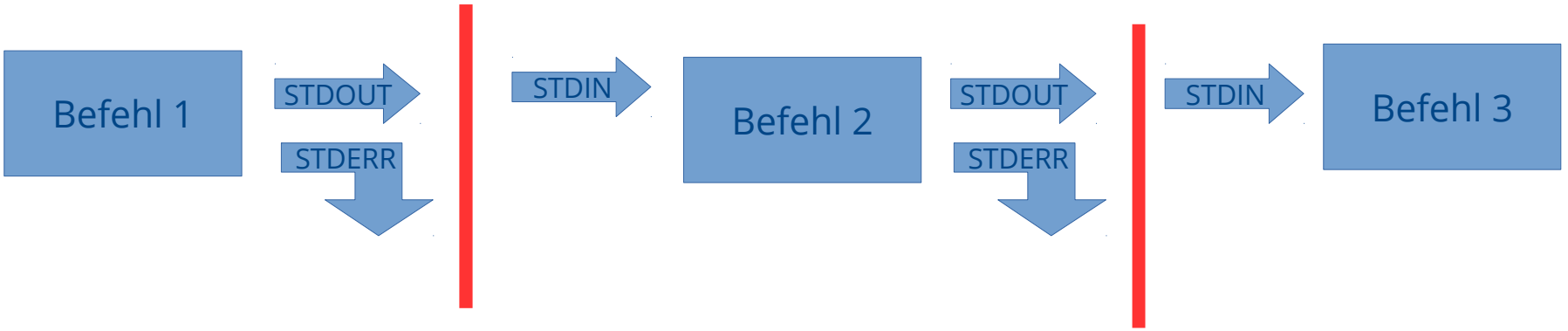
- man kann einfach Zeilen rausfiltern, die mehr als einmal vorkommen
- Befehl **`uniq`**
- *Achtung ! funktionniert nur, wenn die Datei sortiert ist !!*



Man muss also erst die Datei sortieren, dann filtern ...

Befehle kombinieren

- Ein grosser Vorteil von bash : Befehle könne aneinander gereiht werden



- Bash schreibt Ergebnisse in 2 “Kanäle”
  - standard output (STDOUT) → normale Ausgabe
  - standard error (STDERR) → Fehlermeldungen
  - STDOUT kann durch **“Pipe”** (|) zu einem neuen Befehl umgeleitet werden

Beispiel : filtern von Namenlisten

```
carl@tuxbox:material$ cat test.txt
Anna
Paul
Tanja
Klara
Anna
Philip
Klara
Paul
Hanna
Klara
Klara
carl@tuxbox:material$ sort test.txt | uniq
Anna
Hanna
Klara
Paul
Philip
Tanja
carl@tuxbox:material$
```

```
carl@tuxbox:material$ cat test.txt
Anna
Paul
Tanja
Klara
Anna
Philip
Klara
Paul
Hanna
Klara
Klara
carl@tuxbox:material$ sort test.txt | uniq -c
      2 Anna
      1 Hanna
      4 Klara
      2 Paul
      1 Philip
      1 Tanja
carl@tuxbox:material$
```

uniq -c :
zählt, wie oft eine Zeile vorkommt

Übung

- Datei mit Koordinaten aller Exone auf allen humanen Chromosomen
- Wie zähle ich die Exone auf Chromosome 1 ?

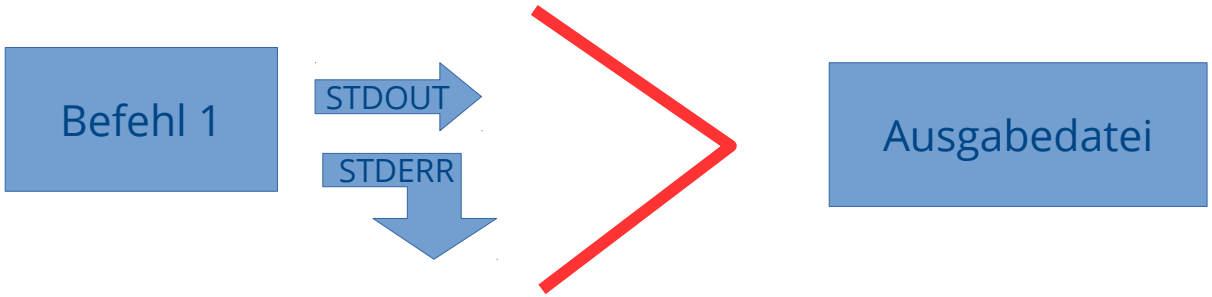
```
$ grep -w chr1 hg19_exon.bed | wc -l
48663

$ grep chr1 hg19_exon.bed | wc -l
244797
```

Warum bekomme ich hier mehr ??

- Wie zähle ich die Exone auf jedem Chromosom einzeln ?

```
$ cut -f1 hg19_exon.bed | sort | uniq -c
  48663 chr1
  21353 chr10
  27087 chr11
  27041 chr12
   8792 chr13
    . . .
```

Ergebnisse in eine Datei schreiben



- Mit dem “>” Zeichen kann man das Ergebnis (STDOUT) eines Befehls in eine Datei schreiben
  - **`Befehl > Datei`** → Datei wir neu angelegt; wenn schon vorhanden, wird überschrieben
  - **`Befehl >> Datei`** → Ergebnis wird an die bestehende Datei angehängt; falls sie noch nicht existiert, wird die Datei angelegt

```
$ cut -f1 hg19_exon.bed | sort | uniq -c > nb_exon_per_chr.txt
```

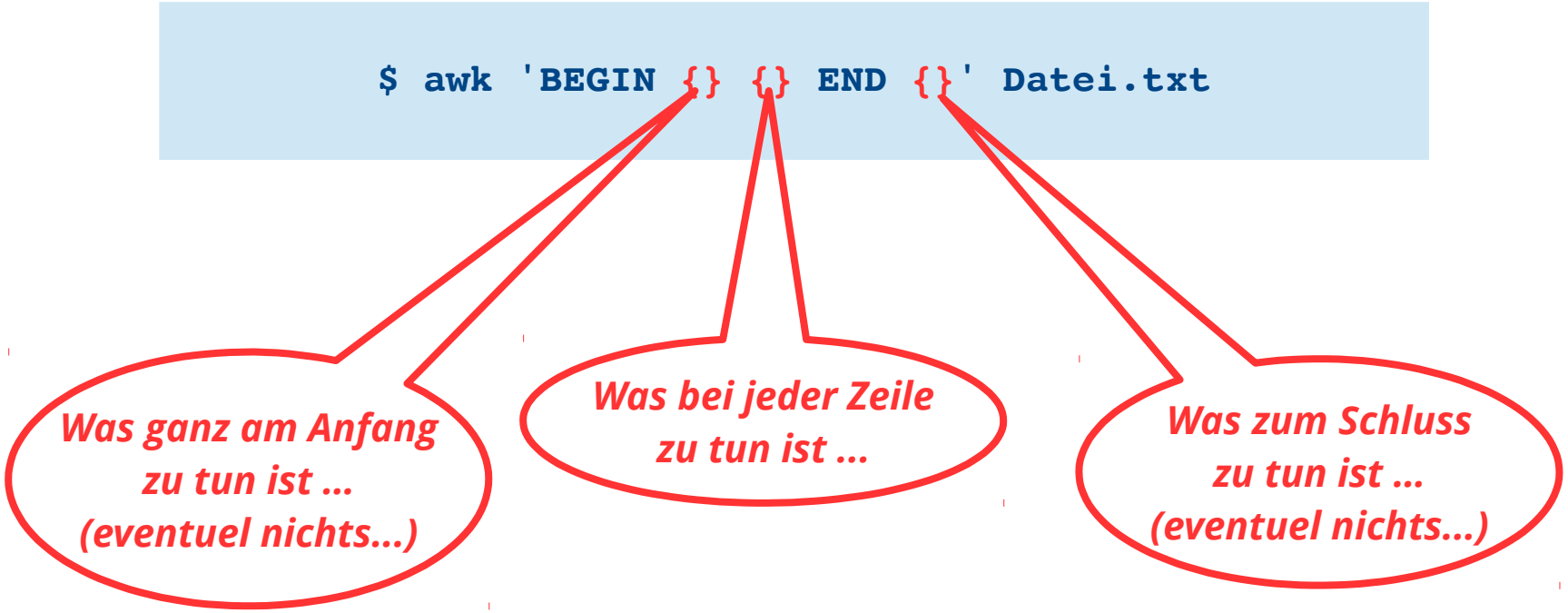
Zusammenfassung

Befehl	Erklärung	Optionen
ls	Inhalt des Verzeichnisses anzeigen	• ls -l → "long" • ls -t → nach Datum sortieren
mkdir	Verzeichnis anlegen	
cd	Verzeichnis ändern	• cd ../ngs02/data → relativ • cd /home/users/ngs02/data → absolut
less	Inhalt einer Datei anschauen	• less -S → lange Zeilen werden NICHT gebrochen
head / tail	Anfang/Ende der Datei anschauen	• head -n 10 → die ersten 10 Zeilen
sort	Datei nach Spalten sortieren	• sort -k 2 → nach der 2. Spalte • sort -k 2 -n → 2. Spalte, numerisch
wc	Zeilen/Zeichen/ Wörter zählen	• wc -l → nur Anzahl Zeilen
cut	Spalte rausschneiden	• cut -f 3 → 3. Spalte • cut -d ";" -f 3 → 3. Spalte, wenn Spalten durch ";" getrennt sind

Weiter mit AWK

- AWK ist eine weitere “Sprache”, die die Möglichkeiten von bash erweitert
- auf den meisten UNIX Systemen verfügbar
- AWK kann Operationen auf allen Zeilen durchführen
- Grundaufbau eines AWK Befehls:

```
$ awk 'BEGIN {} {} END {}' Datei.txt
```



Awk Variablen

- AWK hat wie jede Programmiersprache **Variablen**
- Einige dieser Variablen sind **benutzerdefiniert**
- Andere werden von AWK angelegt und definiert (**spezielle Variablen**); darunter

  - **`FS`** : der Feldtrenner
  - **`OFS`** : der Feldtrenner im Output
  - **`NR`** : die Nummer der aktuellen Zeile die in der Datei bearbeitet wird
  - **`NF`** : Anzahl der Spalten in der Zeile
  - **`$0`** : Inhalt der Zeile
  - **`$1`, `$2`**, ... : Inhalt der Spalten 1,2,... der aktuellen Zeile

Awk Beispiele

```
# Schreibe Spalten 2 und 1 (in der Reihenfolge)

$ awk 'BEGIN {FS="\t"} {print $2,$1}' hg38_exons.bed

# Schreibe Spalten 2 und 1 (in der Reihenfolge) mit
# Tabulierung

$ awk 'BEGIN {OFS= FS="\t"} {print $2,$1}' hg38_exons.bed



# Berechne Ende- Start in einer BED-Datei

$ awk 'BEGIN {OFS= FS="\t"} {print $3-$2}' hg38_exons.bed
```

Awk Beispiele

```
# Berechne Ende- Start in einer BED-Datei
$ awk 'BEGIN {OFS= FS="\t"} {print $3-$2}' hg38_exons.bed
# Berechne die Gesamtlänge
$ awk 'BEGIN {OFS=FS="\t"; L=0} {L=L+$3-$2} END {print L}' hg38_exons.bed
```

Variable L wird am Anfang auf 0 gesetzt

Bei jeder Zeile wird Ende-Start zum aktuellen Wert von L addiert

Am Ende wird der Wert von L angezeigt