Vorlesung Linux-Praktikum

1. Einführung: Dateisystem und erste Befehle

Dirk Frettlöh

Technische Fakultät
Universität Bielefeld

Zusammenfassung heute

`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
`ls -l`	Dateirechte anzeigen
`chmod, chown`	Dateirechte ändern
`umask`	Dateirechte voreinstellen

`.`	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

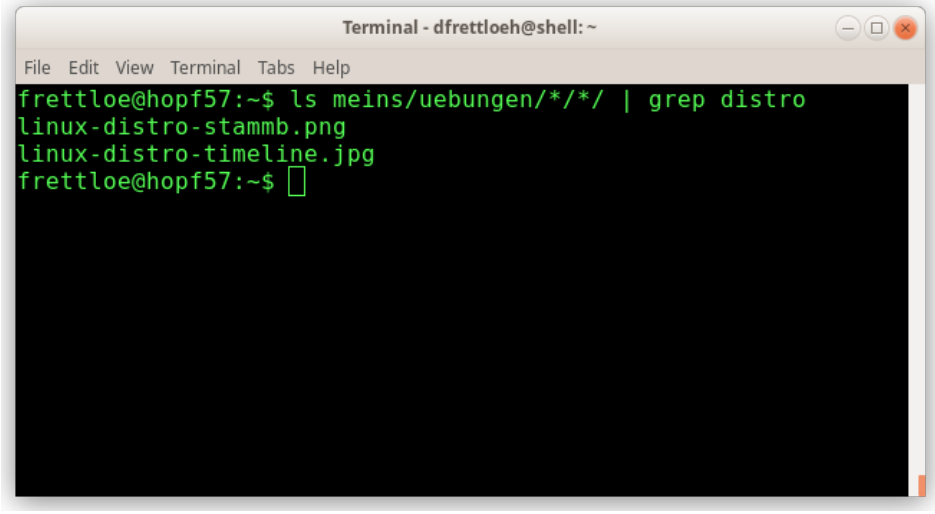
Am Anfang war...

Die Kommandozeile

...aka shell aka terminal.

Die Kommandozeile

(Kommandozeile = Shell = Terminal: das Programm “Terminal”)



Gibt’s auf jeder Linuxkiste und jedem Mac. (Bald? Jetzt?) auch für Windows (“Linux-Subsystem”).

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

Die Kommandozeile

Wozu?

- Wichtiges Werkzeug für alles
- Zeitlos
- Befehle zu mächtigen Werkzeugen kombinieren
- Sollte man als Informatiker kennen
- Sieht cool und kompetent aus

Die Kommandozeile

Wozu?

- Wichtiges Werkzeug für alles
- Zeitlos
- Befehle zu mächtigen Werkzeugen kombinieren
- Sollte man als Informatiker kennen
- Sieht cool und kompetent aus

Wir zeigen das hier für Linux. Bei Macs klappt fast alles genauso, bei Windows braucht man ein "Linux Subsystem". Zu Windows kann ich keine Fragen beantworten. Die Tutoren vielleicht, wenn sie nett sind.

Die Kommandozeile

Wozu?

- Wichtiges Werkzeug für alles
- Zeitlos
- Befehle zu mächtigen Werkzeugen kombinieren
- Sollte man als Informatiker kennen
- Sieht cool und kompetent aus

Wir zeigen das hier für Linux. Bei Macs klappt fast alles genauso, bei Windows braucht man ein "Linux Subsystem". Zu Windows kann ich keine Fragen beantworten. Die Tutoren vielleicht, wenn sie nett sind.

Schreibweise:

Danach "Enter" bzw "Return" drücken.

Zum Beispiel

`$ ls` (zum Ausführen Enter drücken)
`thesis.tex thesis.pdf test.txt`

Der Befehl `ls` zeigt alle Dateien und (Unter-)Ordner im aktuellen Ordner an.

Zum Beispiel

`$ ls` (zum Ausführen Enter drücken)
`thesis.tex thesis.pdf test.txt`

Der Befehl `ls` zeigt alle Dateien und (Unter-)Ordner im aktuellen Ordner an.

`$ pwd` (*print working directory*)

zeigt den aktuellen Pfad an.

Zum Beispiel

`$ ls` (zum Ausführen Enter drücken)
`thesis.tex thesis.pdf test.txt`

Der Befehl `ls` zeigt alle Dateien und (Unter-)Ordner im aktuellen Ordner an.

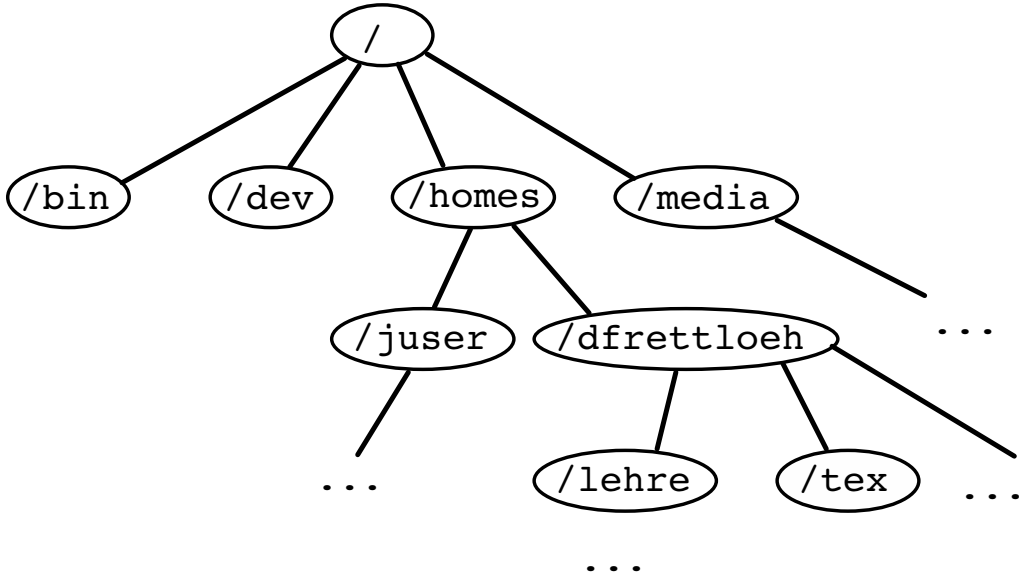
`$ pwd` (*print working directory*)

zeigt den aktuellen Pfad an. Aber was heißt das?

Das Dateisystem

Dateisystem

Das Linux-Dateisystem ist ein Baum

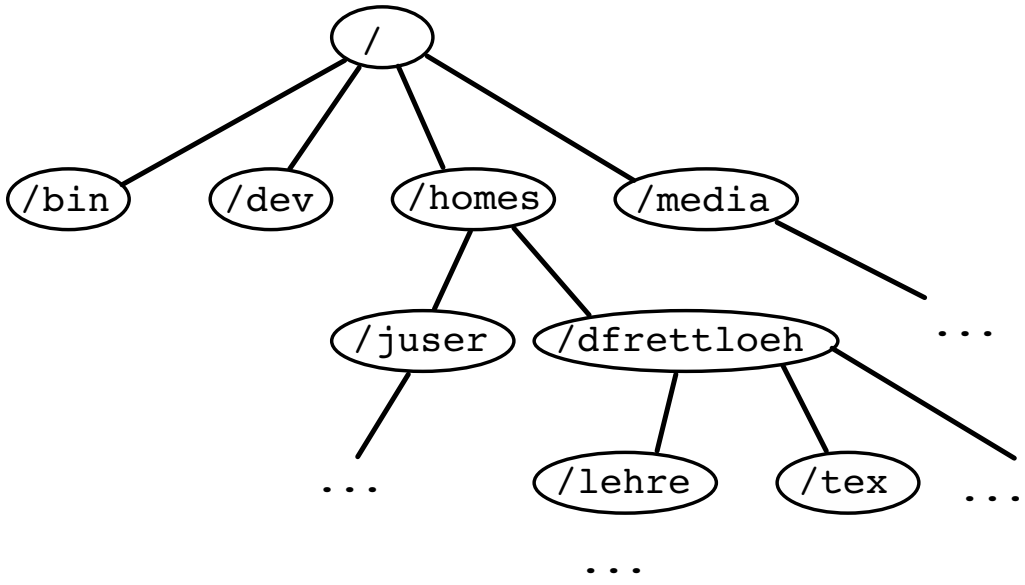


- / : Wurzel
- Verzeichnisse: innere Knoten
- Dateien: Blätter

Pfade: `/homes/dfrettloeh/lehre/unix/`

Dateisystem

Das Linux-Dateisystem ist ein Baum



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

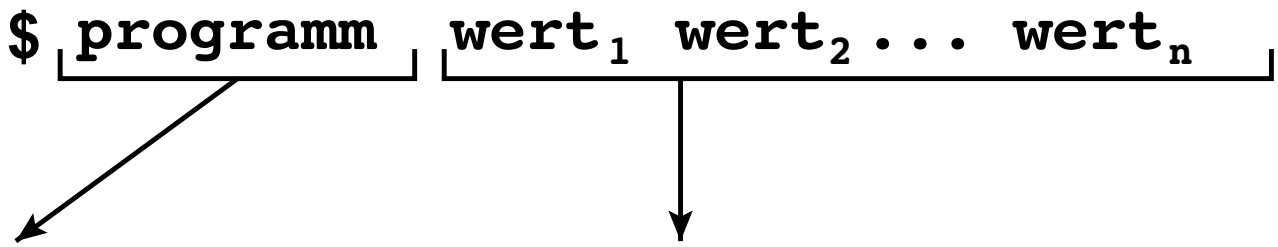
Pfade: `/homes/dfrettloeh/lehre/unix/`

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

Wichtige Befehle

Die Kommandozeile

Erinnerung: Befehl ausführen



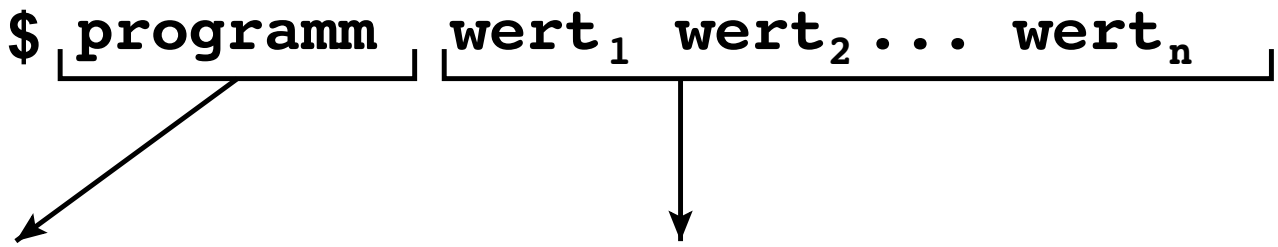
Programmname:
– immer an erster Stelle
– Name muß eindeutig sein

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

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

Die Kommandozeile

Erinnerung: Befehl ausführen



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

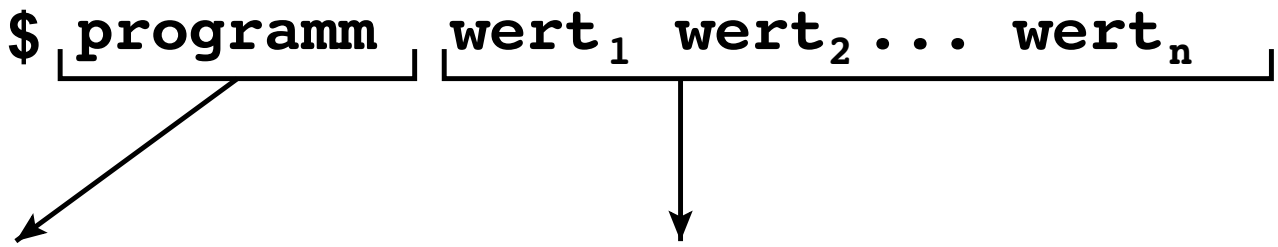
Aufruf–Werte:
- durch Leerzeichen getrennt
– in Anführungszeichen "als ein Wort"
– Interpretation vom Programm abhängig

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

Nützlich: vielen Befehle kann man das Argument `-h` oder `--help` mitgeben. Das zeigt Informationen zum Befehl an.

Die Kommandozeile

Erinnerung: Befehl ausführen



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.

Nützlich: vielen Befehle kann man das Argument `-h` oder `--help` mitgeben. Das zeigt Informationen zum Befehl an.

Man kann natürlich auch alles googeln.

Die Kommandozeile

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:

Die Kommandozeile

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!

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

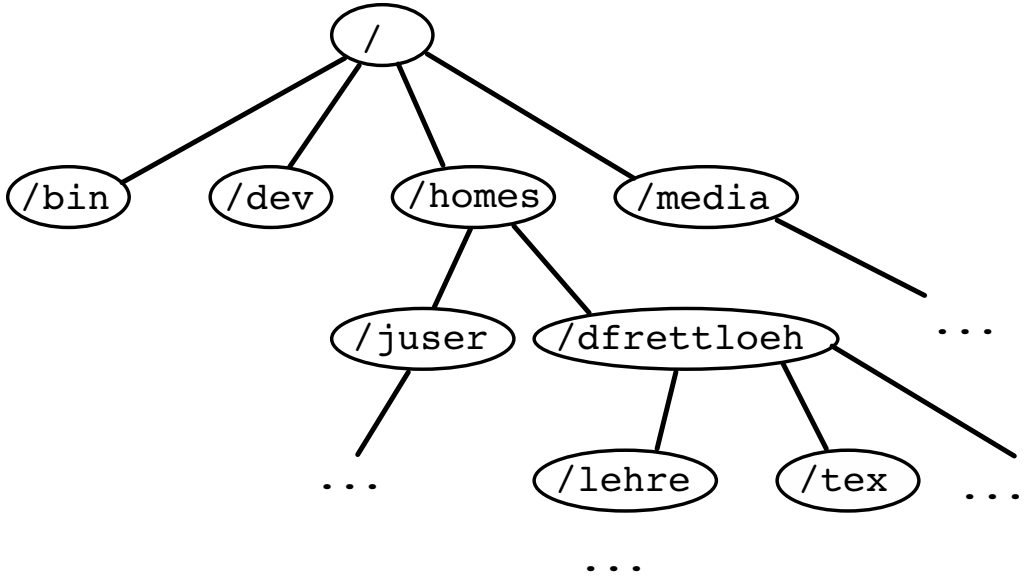
. : Verweis auf das aktuelle Verzeichnis

```
$ 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!

Wildcards

Platzhalterzeichen

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).

Textdateien ≠ **Dokumente**

```
Dokumente sind
keine Textdateien!

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

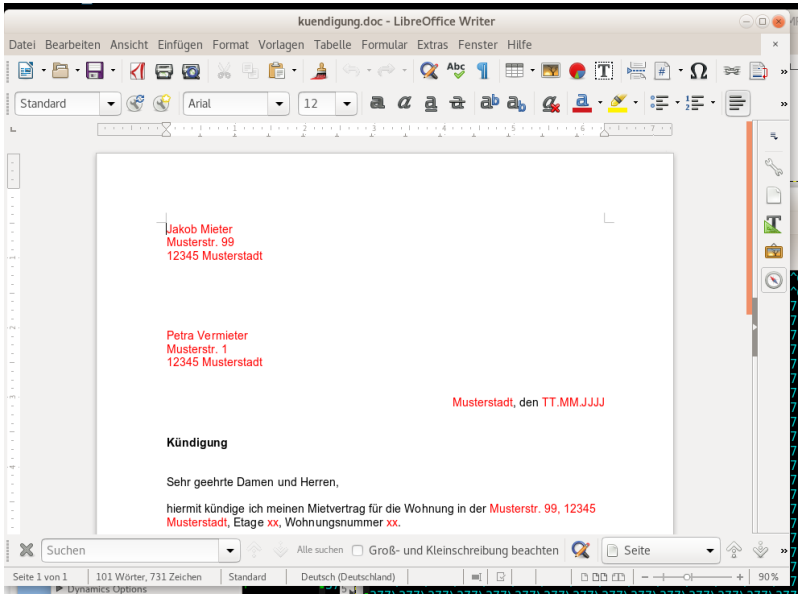
Dokumente sind
keine Textdateien!

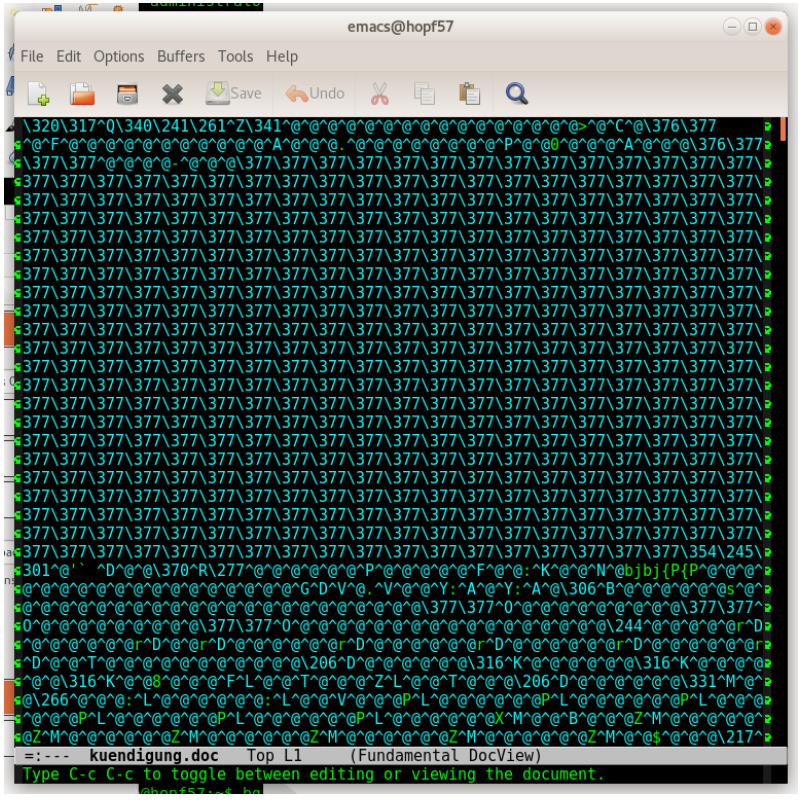
Sie sind
- **Binärdateien** oder wie
- *Programmiersprachen*
  aufgebaut.

Alle Zeichen werden als 8stellige Binärahlen bzw. zweistellige Hexadezimalzahlen kodiert. Früher: ASCII (8 bit), heute UTF8 (8 bit oder mehr).

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.

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

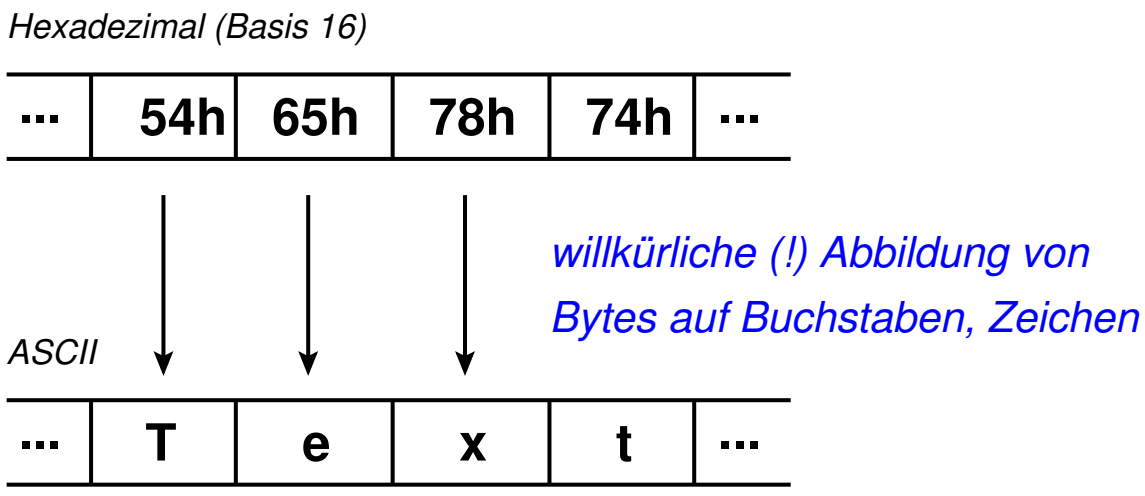
hexdump

Je nach Datei zeigt `more` Schrott an:

```
$ more /dev/random
τ ○ ξ? <??#?
iι⮡□&?p>
```

Dann hilft hexdump.
(Zeigt Bytes einer Datei in Hexadezimal-Kodierung)

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



Dateitypen

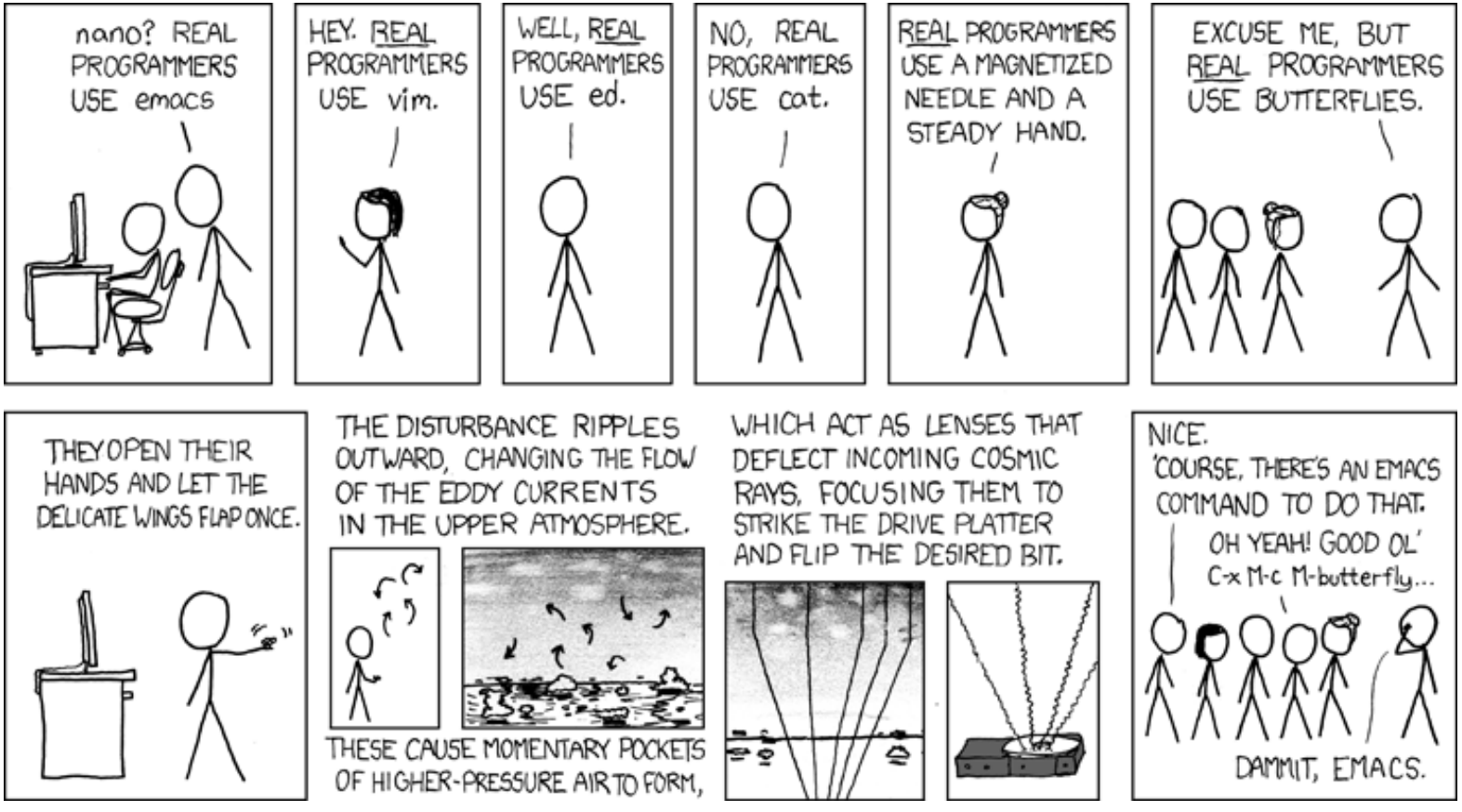
Texteditoren

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

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

Dateitypen

Texteditoren



Dateirechte

...aka *permissions*

Datei- und Verzeichnisrechte

Übersicht

Dateien verwalten

- vieles kennen wir jetzt schon: `pwd`, `ls`, `cd`, `cp`, `mv`, `rm`

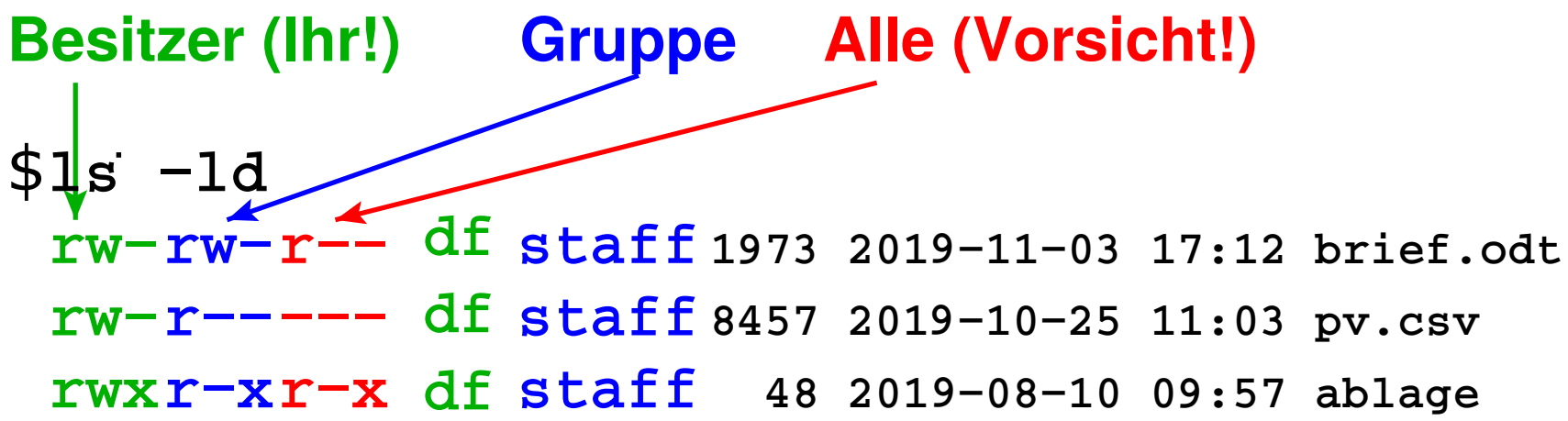
Weitere typische Aufgaben

- Lese- / Schreibrechte verstehen
- ... und verwalten

Datei- und Verzeichnisrechte

Zugriffsrechte

3-stufiges System von Berechtigungen:



Grundlegende Berechtigungen (engl. *permissions*):

r	read	Öffnen / Lesen erlaubt
w	write	Schreibzugriff / Löschen erlaubt
x	execute	Dateien: Programmausführung erlaubt
		Verzeichnisse: Durchgreifen erlaubt

Datei- und Verzeichnisrechte

Zu welchen Gruppen gehöre ich?

groups

```
$ groups
bmstaff teachlinux vlvkinf tak ...
```

id (identity)

```
$ id
uid=22227(df) gid=12000(bmstaff) groups=...
```

- gid: primäre Gruppe

Wann bekommt man zusätzliche Gruppen? (Im Techfak-System)

- Maschinenbezogen (z.B. *audio* bei lokalem login an PCs)
- Statuswechsel (HiWi werden, Bachelorarbeit schreiben)

Datei- und Verzeichnisrechte

Berechtigungen ändern

chmod (change file mode)

`$ chmod g-w datei`	kein Schreibzugriff für Gruppe
`$ chmod u+w datei`	erlaube Schreibzugriff für sich selbst
`$ chmod o=r datei`	erlaube nur Lesezugriff für alle (w,x werden gelöscht)
`$ chmod go-rwx *.txt`	für *.txt-Dateien alle Zugriffe für Gruppe und alle wegnehmen
`$ chmod g=rw,o= datei`	Gruppe darf lesen und schreiben, andere haben keinen Zugriff

u : Berechtigung für Besitzer (user; erster rwx-Block)
g : Berechtigung für Gruppe (group; zweiter rwx-Block)
o : Berechtigung für Alle (other; dritter rwx-Block)

Datei- und Verzeichnisrechte

Berechtigungen ändern

chown (change owner)

Im Prinzip so:

```
$ chown juser scan0003.pdf
chown:  changing ownership of 'scan0003.pdf':
Operation not permitted
```

Das darf nur der Superuser (su)

```
$ sudo chown juser datei.txt
```

Datei- und Verzeichnisrechte

Prioritäten auf Dateiberechtigungen

Die speziellste anwendbare Berechtigung gilt:
(am Beispiel jeweils aus Sicht des Nutzers `df`)

Berechtigung	Nutzer	Gruppe	df darf lesen
`-r--------`	df	staff	ja
`----r--r--`	df	staff	nein
`----r--r--`	juser	staff	ja
`-------r--`	juser	staff	nein
`-------r--`	nn	nn	ja

(df sei Mitglied der Gruppe `staff`, aber nicht in `nn`)

Datei- und Verzeichnisrechte

Berechtigungen auf Verzeichnissen

'w'-Berechtigung auf Verzeichnis

- Anlegen von Dateien / Unterverzeichnissen
- Löschen von Dateien / Unterverzeichnissen

Zusammenspiel von Datei- und Verzeichnisberechtigungen

```
$ ls -ld
 dr-xr-xr-x  4  df    staff  ...  .
 drwxrwxr--  3  root  root   ...  ..
 -rw-r--r--  1  df    staff  ...  brief.txt
```

- `brief.txt` kann verändert werden (Dateiberechtigung)
- `brief.txt` kann *nicht* gelöscht werden (Verzeichnisber.)
- Neue Dateien können *nicht* angelegt werden (Verzeichnisberechtigung)

Datei- und Verzeichnisrechte

Berechtigungen auf Verzeichnissen

'w' auf Gruppenverzeichnis hebelt Datei-Schreibschutz aus

```
drwxrwxr-x  4  df    projekt  ...  .
drwxrwxr--  3  root  root     ...  ..
-rw-r--r--  1  df    projekt  ...  brief.txt
```

Nutzer `nn` sei ebenfalls in der Gruppe `projekt`:

- nn kann `brief.txt` nicht editieren, aber
- nn kann `brief.txt` löschen und neu anlegen

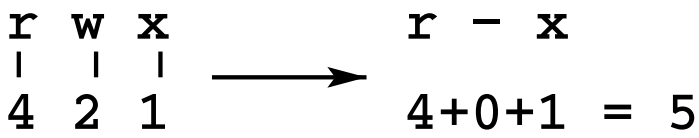
Folgerung:

- *Niemals* das Home-Verzeichnis gruppen-/weltschreibbar machen!

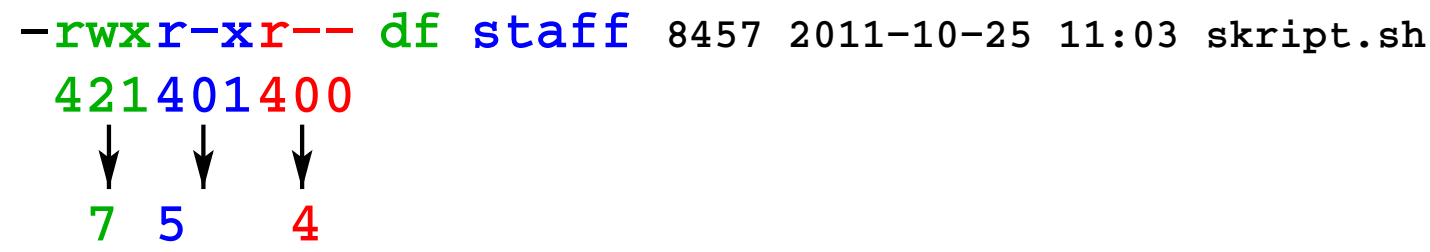
Datei- und Verzeichnisrechte

Nerd-Variante

Bitweise Kodierung: Zahlen für gesetzte Berechtigungen addieren



Beispiel:




```
$ chmod 754 skript.sh
```

Erste Ziffer für User, zweite Ziffer für Group, dritte Ziffer für Others

Datei- und Verzeichnisrechte

Berechtigungen auf Verzeichnissen

'r'-Berechtigung auf Verzeichnis

- erlaubt *Dateinamen* zu lesen (und sonst nichts!)

'x'-Berechtigung auf Verzeichnis

- erlaubt *Inhalt* von Dateien und Unterverzeichnissen zu lesen

Typischerweise: `rx` zusammen setzen oder wegnehmen

Datei- und Verzeichnisrechte

umask

Jeweils von Hand Rechte ändern ist lästig.

Rechte für neu angelegte Dateien voreinstellen mit `umask`.

Logik genau andersum:

$$\begin{array}{r} 777 \\ -\,023 \\ \hline 754 \end{array}$$

`umask 023` bewirkt also für alle neuen Dateien 754:

`rwxr-xr--`

Datei- und Verzeichnisrechte

umask

Jeweils von Hand Rechte ändern ist lästig.

Rechte für neu angelegte Dateien voreinstellen mit `umask`.

Logik genau andersum:

$$\begin{array}{r} 777 \\ -\,023 \\ \hline 754 \end{array}$$

`umask 023` bewirkt also für alle neuen Dateien 754:
`rwxr-xr--`

(Lies "Maske" als Filter.)

Die meisten Programme erzeugen neue Dateien *ohne* das x-bit.
(Ausnahme z.B. Compiler, die sollen ja eine ausführbare Datei liefern)

Datei- und Verzeichnisrechte

setuid und setgid

Auf aktuellen Linux-Rechnern sieht man oft so was:

```
drwsrwSr--
```

- Das User s: `setuid` führe solche Dateien mit den Rechten seines Besitzers aus (nicht mit denen des Users, der sie startet.)
- Das Groups s: `setgid`
  - Führe solche *Dateien* mit den Rechten seiner Gruppe aus (nicht mit der des Users, der sie startet.)
  - Neu angelegte Dateien in solchen *Ordnern* erben die Rechte dieses Ordners (nicht die der `umask` des Users, der sie erzeugt)

Datei- und Verzeichnisrechte

setuid und setgid

Auf aktuellen Linux-Rechnern sieht man oft so was:

```
drwsrwSr--
```

- Das User s: `setuid` führe solche Dateien mit den Rechten seines Besitzers aus (nicht mit denen des Users, der sie startet.)
- Das Groups s: `setgid`
  - Führe solche *Dateien* mit den Rechten seiner Gruppe aus (nicht mit der des Users, der sie startet.)
  - Neu angelegte Dateien in solchen *Ordnern* erben die Rechte dieses Ordners (nicht die der `umask` des Users, der sie erzeugt)

Großes S: das x-Bit ist nicht gesetzt, aber das setuid/setgid Bit schon. (mehr: 'sticky bit' suchen)

Etwas sinnlose Einstellung, da die Gruppe nicht in den Ordner kann.

Zusammenfassung

`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
`ls -l`	Dateirechte anzeigen
`chmod, chown`	Dateirechte ändern
`umask`	Dateirechte voreinstellen

`.`	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

Ende der heutigen Vorlesung

Sie probieren das alles gleich in den Tutorien aus.

Viel Spaß! Bis morgen!