

# Vorbereitungskurs Informatik – Teil 1

FSI Informatik

Uni Erlangen-Nürnberg

16. Oktober 2015

# Allgemeines

## Wer sind wir?

- Fachschaftsinitiative (kurz FSI)
  - Informatik
  - Informations- und Kommunikationstechnik
  - Computational Engineering
  - Wirtschaftsinformatik
  - Technomathematik
  
- Was machen wir?
  - Erstsemestereinführung
  - Bereitstellen von Prüfungsfragen und weiteren Infos
  - Genereller Ansprechpartner für Studenten
  - Sommerfest
  - Vertretung der studentischen Interessen in Gremien
  - Was noch so anfällt...

### Dienstag 13.10.2015

<b>Zeit</b>	<b>Raum</b>	<b>Inhalt</b>
09:00–11:00	H16	Vorlesung (Teil 1)
11:00–13:00	CIP-Pools	Übung (Teil 1)
Mittagspause		
14:00–16:00	H16	Vorlesung (Teil 2)
16:00–18:00	CIP-Pools	Übung (Teil 2)

# Der Erlanger Informatiker



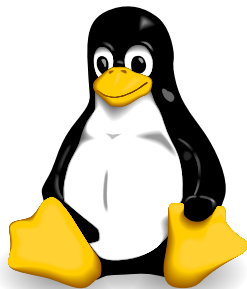
Nicht so wirklich.



Auch eher nicht.



! Hacker, Frickler, Ingenieur :-)



# Allgemeines

## Linux – was ist das?

- Eigentlich nur ein Betriebssystemkern
- Meistens meint man mit *Linux* eine Zusammenstellung von:
  - Betriebssystem
  - (Arbeits-)Programmen
- Diese *Linux-Distributionen* haben eigene Namen und Versionsnummern, z. B.:



- **debian** (hier im CIP installiert)

- **ubuntu**<sup>®</sup>



- **openSUSE**
-  **Gentoo**

- ...



# Allgemeines

Wie schaut's im CIP aus?

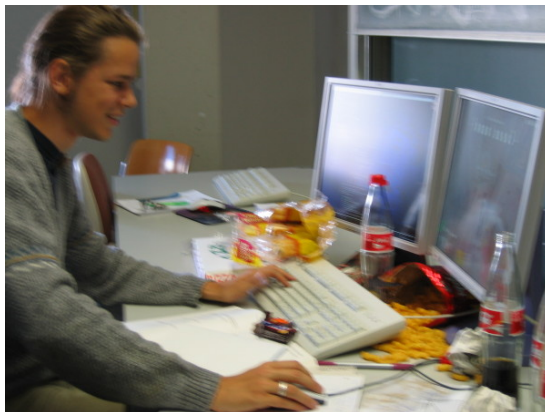
CIP-Pools im Blauen Hochhauses:

- Linux-Arbeitsrechner
- Drucker
- Farbdrucker-Scanner-Multifunktions-Monster (im CIP 2)



# Allgemeines

Warnung!



Essen und Trinken verboten!  
(Loginentzug droht)

# Allgemeines

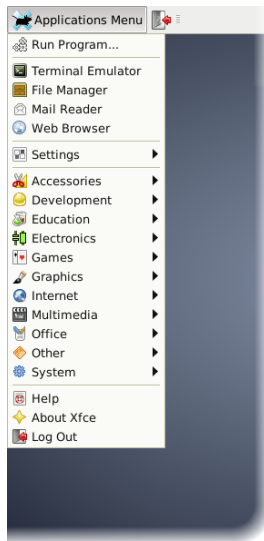
## Window-Manager – XFCE

### Window-Manager

Bestimmt Aussehen und Verhalten der grafischen Oberfläche



- Gut geeignet für den Einstieg
- Thunar (Dateimanager)
- Iceweasel – entspricht Firefox
- *System*-Menü zur Konfiguration
- Übersichtliche schlanke Oberfläche





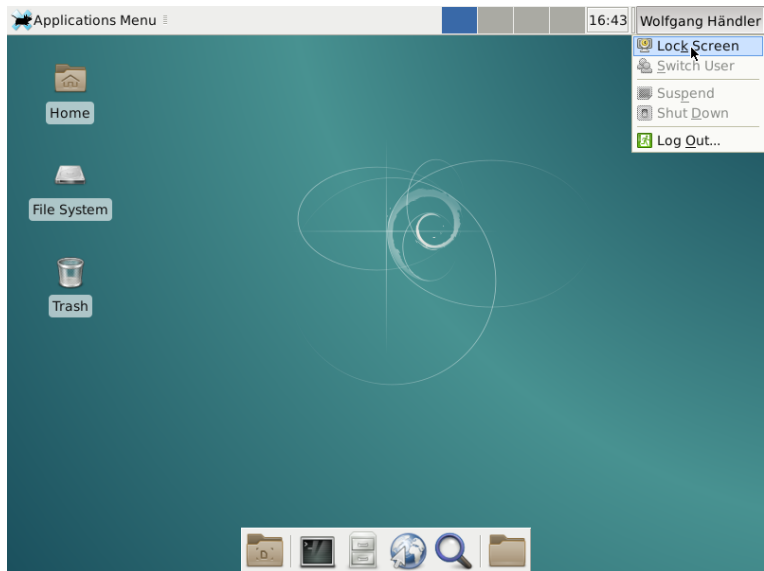
- Intuitive Bedienung („ähnlich wie unter Windows“)
- In der Standardeinstellung komplett auf englisch – aber das solltet ihr alle können. . .
- Wir trauen euch zu, dass ihr selbstständig zurecht kommt :-)
- Daher: in diesem Kurs Konzentration auf Befehlszeile & Co.

Gibt's trotzdem Probleme?

Universeller Lösungsalgorithmus: <https://xkcd.com/627/>

# Allgemeines

## Bildschirm sperren und abmelden



# Allgemeines

## Befehlszeile – Warum?

A screenshot of a terminal window. The title bar reads 'el79irih:~'. The prompt is 'el79irih@fau001 ~\$'. The command entered is 'echo "Hello World"'. The output is not visible, only the prompt and command are shown.

```
el79irih:~  
el79irih@fau001 ~$ echo "Hello World" |
```

Getippte Befehle anstelle grafischer Anwendungen.

**Warum?! Ist das nicht ein riesiger Rückschritt?**

# Allgemeines

## Verkleinern eines Bildes

### Beispiel: Verkleinern eines Bildes

- 1 Grafikprogramm aus dem Startmenü ausführen.
- 2 *Datei* → *Öffnen* klicken.
- 3 Den richtigen Ordner suchen.
- 4 Die Bilddatei auswählen.
- 5 Im *Bild*-Menü auf den Befehl *Skalieren* klicken.
- 6 Die neue Größe eingeben.
- 7 *Datei* → *Speichern unter* klicken.
- 8 Den neuen Dateinamen eingeben.



# Allgemeines

## Verkleinern eines Bildes

Und auf der Befehlszeile?

Wenn man erst einmal weiß wie, genügt ein Befehl<sup>1</sup>:

```
$ convert -resize 300 gnu.png gnu-klein.png
```

Und das kann man auch **mit einem einzigen Befehl** für 100 Dateien durchführen!

- Zwar höhere Einarbeitungszeit. . .
- . . . aber auf Dauer deutlich schneller!

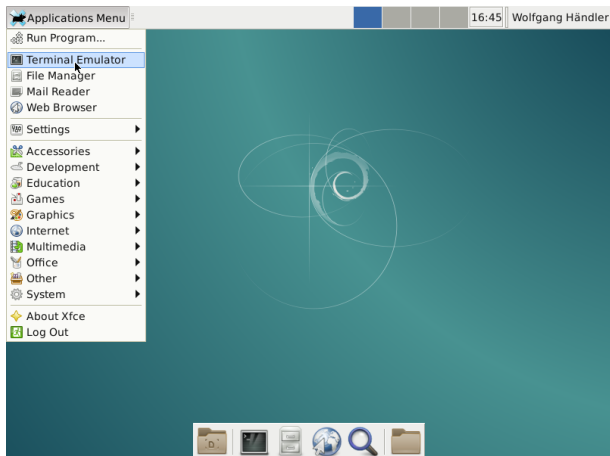
---

<sup>1</sup>\$ ist das sogenannte *Prompt*-Symbol und muss nicht mit eingetippt werden.

# Terminal

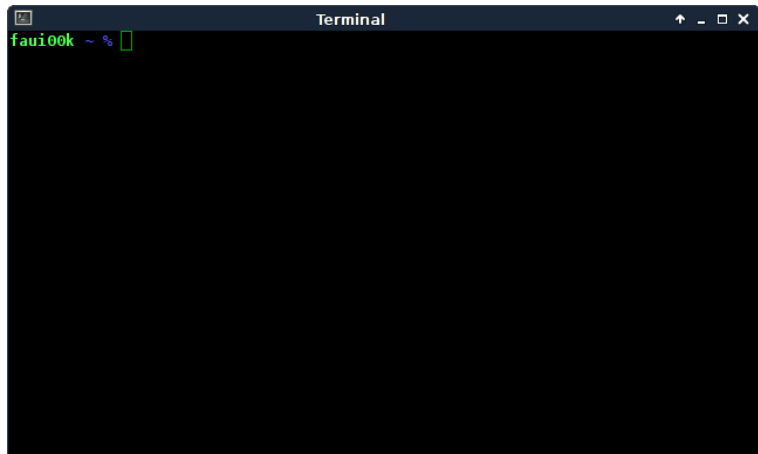
Und wo kann ich diese Befehle eingeben?

Das passende Programm von XFCE, der Standard-Desktop-Umgebung im CIP, heißt *Terminal*:



# Terminal

... und sieht so aus:



### Shell

- Programm, welches eingetippte Befehle entgegennimmt
- **bash** ist die Standardshell im CIP



Im Terminal kann man jetzt Befehle eingeben:

```
$ echo
```

echo gibt den übergebenen Text unverändert wieder aus.

# Befehlsaufbau

## Befehle mit einem Parameter

Dazu brauchen wir Parameter:

### Muster

*<Befehl> <Parameter>*

```
$ echo foo  
foo
```

# Befehlsaufbau

## Mehrere Parameter

Also einmal mit zwei Wörtern:

```
$ echo foo bar  
foo bar
```

... und noch ein paar Leerzeichen mehr:

```
$ echo foo    bar  
foo bar
```

# Befehlsaufbau

## Quoting

### Problem:

```
$ echo foo      bar
foo bar
```

Mehrere Parameter werden durch Leerzeichen getrennt – wie viele Leerzeichen, spielt keine Rolle.

Durch *Quoting* kann man die Spezialbedeutung von Leerzeichen<sup>2</sup> aufheben – der Text, der in Anführungszeichen steht, wird als ein einziger langer Parameter interpretiert.

### Lösung:

```
$ echo 'foo      bar'
foo      bar
```

<sup>2</sup>und anderen Sonderzeichen

Je nach Befehl können auch verschiedene Optionen angegeben werden, um das Verhalten des Befehls zu verändern:

### Muster

```
<Befehl> <Optionen> <Parameter>
```

Bei `echo` bewirkt die Option `-n`, dass nach der Ausgabe keine neue Zeile angefangen wird.

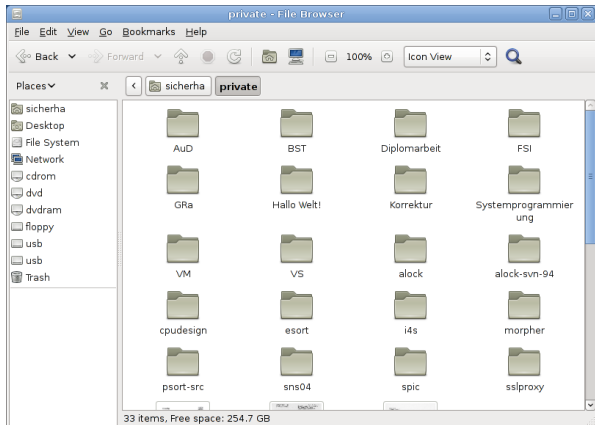
```
$ echo -n foo  
foo $ _
```

# Herumklettern im Dateisystembaum

Hilfe! Wo ist der Explorer?

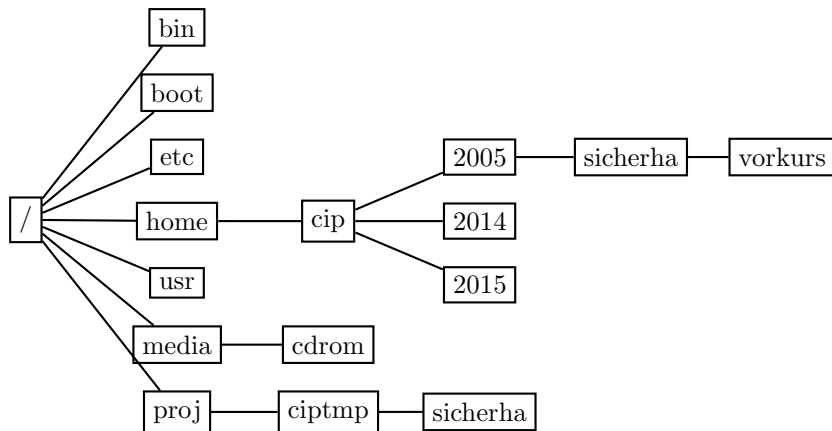
Noch schnell: grafische Dateibrowser für den Notfall:

- Nautilus
- Dolphin
- Thunar
- ...



# Herumklettern im Dateisystembaum

## Aufbau des Verzeichnisbaums



# Herumklettern im Dateisystembaum

## Unterschiede zu Windows

- Es gibt nur einen großen Dateisystembaum, nicht mehrere mit jeweils einem Laufwerksbuchstaben.
- Pfadtrenner: / (*Slash*) statt \ (*Backslash*).
- Zwischen Groß- und Kleinschreibung wird unterschieden!



# Herumklettern im Dateisystembaum

mount – CDs und DVDs

`mount <path>`

<code>mount /media/dvd</code>	hängt DVDs ein (Dateisystem <i>udf</i> )
<code>mount /media/cd</code>	hängt CDs oder DVDs ein ( <i>iso9660</i> )

Bevor das Laufwerk sich wieder **öffnen** lässt, muss es wieder ausgehängt werden.

`umount <path>`

<code>umount /media/dvd</code>	hängt DVDs aus
<code>umount /media/cd</code>	hängt CDs oder DVD aus

# Herumklettern im Dateisystembaum

mountusb – USB-Sticks einhängen

## USB im CIP

mountusb

hängt den USB-Stick unter /media/usb ein

umountusb

hängt den USB-Stick wieder aus

## Anmerkungen

- Vor dem Abziehen des Sticks unmounten → sonst Datenverlust!

# Herumklettern im Dateisystembaum

Wo zum Teufel sind wir überhaupt?

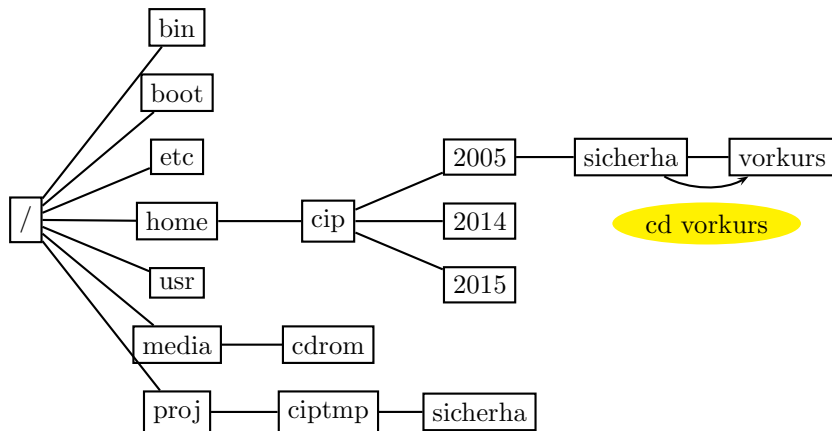
```
pwd
```

pwd (*print working directory*) gibt das aktuelle Verzeichnis aus.

```
$ pwd  
/home/cip/2005/sicherha
```

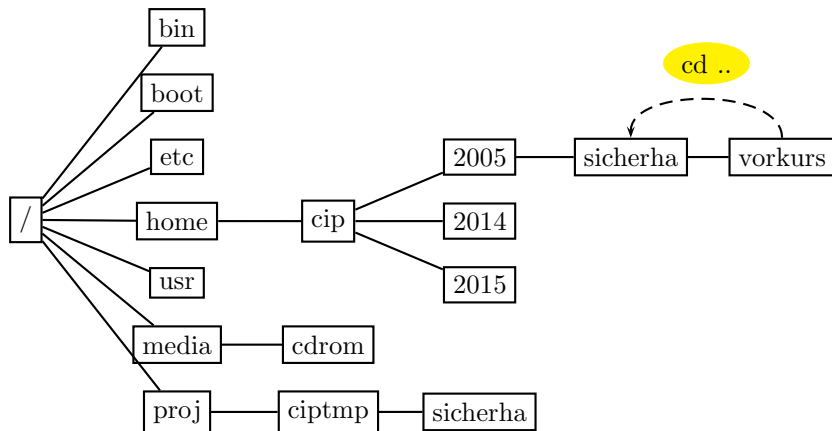
# Herumklettern im Dateisystembaum

## Verzeichniswechsel



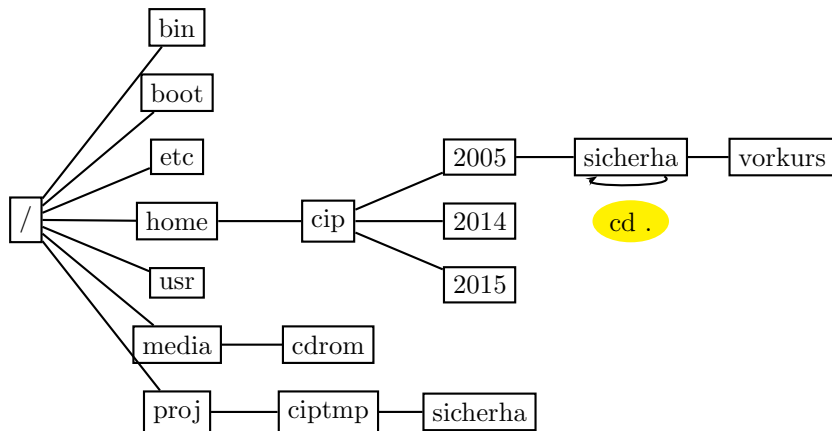
# Herumklettern im Dateisystembaum

Verzeichniswechsel ins übergeordnete Verzeichnis



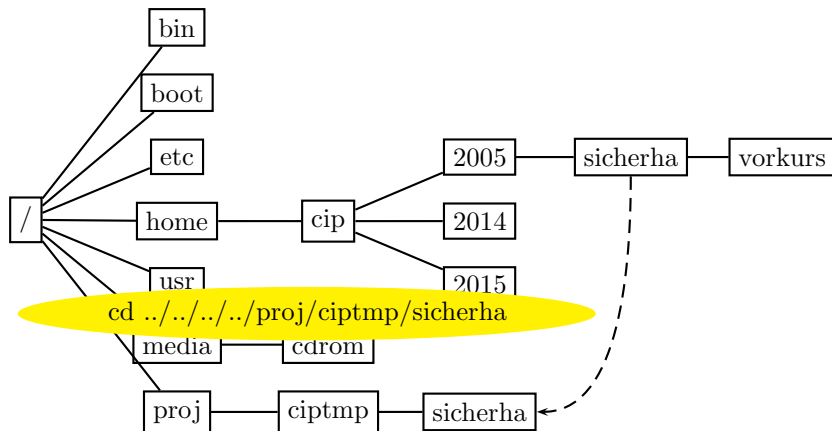
# Herumklettern im Dateisystembaum

„Verzeichniswechsel“



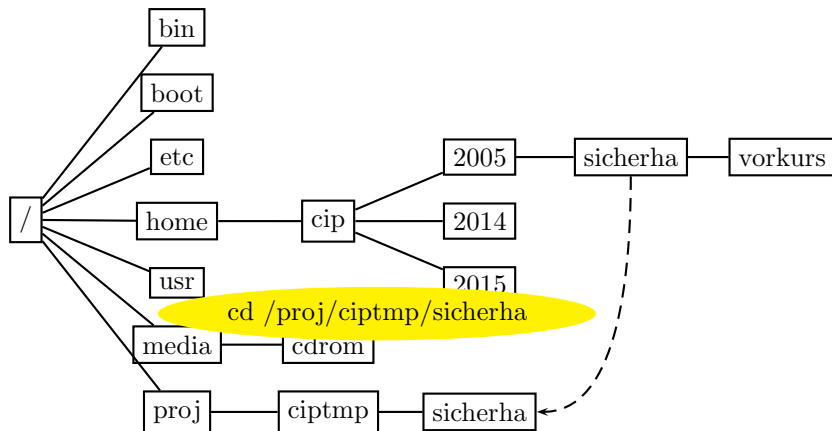
# Herumklettern im Dateisystembaum

Relativer Verzeichniswechsel (relativ zum aktuellen Verzeichnis)



# Herumklettern im Dateisystembaum

Absoluter Verzeichniswechsel (ausgehend vom Wurzelverzeichnis – vorangestellter /)





# Herumklettern im Dateisystembaum

## Verzeichniswechsel

### cd

Mit `cd` (= *change directory*) wechselt man zwischen Verzeichnissen.

### Beispiele

- `cd bin` – wechselt in das Unterverzeichnis 'bin' im aktuellen Verzeichnis (*relativer Pfadwechsel*)
- `cd /bin` – geht in das Verzeichnis 'bin' unterhalb des Root-Verzeichnisses / (*absoluter Pfadwechsel*)
- `cd ..` – wechselt eine Verzeichnisebene nach oben
- `cd ../testy` – wechselt eine Verzeichnisebene nach oben **und** darin in das Verzeichnis 'testy'
- `cd` – geht in das *Home*-Verzeichnis
- `cd -` – geht in das letzte besuchte Verzeichnis

# Herumklettern im Dateisystembaum

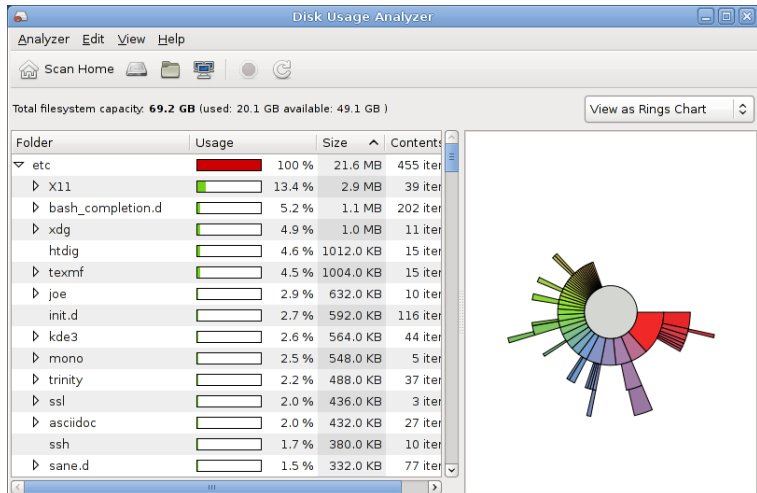
## Home und ciptmp

- Jeder Benutzer besitzt ein *Home*-Verzeichnis (`/home/cip/2015/<userlogin>`):
  - Es steht nur begrenzter Speicherplatz zur Verfügung (500 MB)
  - Dort liegen Konfigurationen und Nutzdaten
  - Der Inhalt wird täglich gesichert und ist zentral gespeichert, also auf allen Rechnern gleich
  - Kurzschreibweise fürs *Home*-Verzeichnis: `~` (*Tilde-Zeichen*)
- Mehr Speicherplatz ist im *ciptmp* verfügbar (`/proj/ciptmp/<userlogin>`):
  - Wird nicht gesichert und kann ohne Vorwarnung gelöscht werden!
  - Wird erst bei Betreten eingebunden (d. h. ein `ls` auf `/proj/` kann u. U. den Anschein erwecken, dass das Verzeichnis leer ist!)

# Herumklettern im Dateisystembaum

Speicherplatzverbrauch – grafisch mit baobab

```
$ baobab /etc/
```



# Herumklettern im Dateisystembaum

Speicherplatzverbrauch – per Konsole

du

Mit `du` (= *disk usage*) kann man sich den Speicherplatz anzeigen lassen.

## Beispiele

- `du` – gibt den Speicherbedarf aller Dateien aus  
(rekursiv für jeden Ordner)
- `du -h` – *-h = human-readable*  
→ gibt die Größen besser lesbar aus
- `du -sh` – gibt den Speicherbedarf des aktuellen Ordners lesbar aus

# Inhalte aufzeigen

## Verzeichnisinhalt

ls

ls listet den Inhalt eines Verzeichnisses auf.

### Beispiele

- ls – listet Inhalt des aktuellen Verzeichnisses auf
- ls verzeichnis – listet Inhalt des angegebenen Verzeichnisses auf
- ls -d verzeichnis – gibt Informationen zum angegebenen Verzeichnis aus (nicht aber den Inhalt)
- ls -l – ausführliche Verzeichnisaufstellung (Dateigrößen, Rechte, Zeitstempel etc.)
- ls -a – listet auch versteckte Dateien (Dateien, die mit einem Punkt beginnen) auf

# Inhalte aufzeigen

## Beispiele

### Normales `ls` vs. `ls -a`

```
$ ls  
a.txt mein_bild.jpg
```

```
$ ls -a  
.  ..  .bash_history  a.txt  mein_bild.jpg
```

- `ls -a` zeigt wirklich alle Einträge des Verzeichnisses an!
- Einträge, die mit einem „.“ beginnen, werden normalerweise als „müssen nicht immer sichtbar sein“ interpretiert und versteckt, z. B.:
  - „.“ ist immer das aktuelle Verzeichnis
  - „..“ ist immer das übergeordnete Verzeichnis
  - „.bash\_history“ enthält z. B. Befehle, die früher eingegeben wurden

# Fahrt aufnehmen

## Tab-Vervollständigung

Mit einem Druck auf <TAB> wird u. a. Folgendes ergänzt:

- Namen von Befehlen
- Datei- und Verzeichnisnamen

```
$ ls
Desktop  folien_vorkurs_2015_tag1.pdf

$ file fo<TAB>
$ file folien_vorkurs_2015_tag1.pdf
folien_vorkurs_2015_tag1.pdf: PDF document, version 1.4
```

# Fahrt aufnehmen

## Tab-Vervollständigung

Bei nicht eindeutiger Eingabe zeigt ein weiterer Druck auf <TAB> eine Liste von möglichen Alternativen an:

```
$ ls
folien_vorkurs_2015_tag1.pdf    folien_vorkurs_2015_tag2.pdf
vortrag_vorkurs_2014.pdf       vortrag_vorkurs_2012.pdf

$ file f<TAB>
$ file folien_vorkurs_2015_tag<TAB><TAB>
folien_vorkurs_2015_tag1.pdf  folien_vorkurs_2015_tag2.pdf
$ file folien_vorkurs_2015_tag2.pdf
folien_vorkurs_2015_tag2.pdf: PDF document, version 1.4
```



# Fahrt aufnehmen

(Bestimmte) Befehle wiederholen

- Mit Cursortasten hoch/runter durch letzte Befehle bewegen
- Mit !<Befehl> letzten Befehl mit Namen <Befehl> ausführen

```
$ file folien_vorkurs_2015_tag2.pdf
folien_vorkurs_2015_tag2.pdf: PDF document, version 1.4
```

...andere Befehle (nur nicht file) eingeben ...

```
$ !file
file folien_vorkurs_2015_tag2.pdf
folien_vorkurs_2015_tag2.pdf: PDF document, version 1.4
```

# Fahrt aufnehmen

## Suche in der Befehlshistory

- `Ctrl-R` liefert den Modus „reverse-i-search“.
- Tippt man nun den Teil eines Befehls ein, erscheint der zuletzt benutzte Befehl, der diesen Teil enthält.
- Durch nochmaliges Drücken von `Ctrl-R` kann man durch mögliche Befehle scrollen.
- Hat man gefunden, was man sucht, kann man den Befehl noch beliebig editieren (Pfeiltaste zur Navigation) und dann ausführen.

# Fahrt aufnehmen

## Copy & Paste in Terminals

**copy:** Den Text, den man kopieren will, einfach markieren. . .

**paste:** . . . und an der gewünschten Stelle mit einem Klick auf das Mausrad (oder mit `Shift-Insert`) einfügen.

# Elementare Befehle

man-pages – das Hilfesystem unter Unix

## Typische Verwendung

```
man <Befehl>
```

### man echo

```
ECHO(1)                                User Commands                                ECHO(1)
```

#### NAME

```
echo - display a line of text
```

#### SYNOPSIS

```
echo [OPTION]... [STRING]...
```

#### DESCRIPTION

```
Echo the STRING(s) to standard output.
```

```
-n      do not output the trailing newline
```

# Elementare Befehle

Bedienung von `man`

## Die wichtigsten Tasten

- **Scrollen (zeilenweise)**: Pfeiltaste hoch/runter
- **Scrollen (seitenweise)**: Bild auf/ab
- **Suchen**: /suchbegriff<ENTER>
- **Nächster Treffer**: n
- **Vorheriger Treffer**: N
- **Beenden**: q

Tipp: Auch andere Befehle wie `less` lassen sich so bedienen!

# Elementare Befehle

Und wenn ich gar nicht weiß, welchen Befehl ich brauche?

*apropos* ist dein Freund!

`apropos <Suchbegriff>`

```
$ apropos rename
```

```
...
```

```
mv (1) - move (rename) files
```

```
prename (1) - renames multiple files
```

```
rename (2) - change the name or location of a file
```

```
...
```

Wenn die Anzeige zu lang wird, hilft `apropos <Befehl> | less` weiter.

# Elementare Befehle

mv – Verschieben

## Aufbau

`mv <Quelle> <Ziel>`

## Beispiele

`mv alt neu` – benennt die Datei 'alt' in 'neu' um  
(geht auch für Verzeichnisse)

`mv foo dinge/` – verschiebt die Datei 'foo' aus dem aktuellen  
Verzeichnis in das Verzeichnis 'dinge'

# Elementare Befehle

cp – Kopieren

## Aufbau

`cp <Quelle> <Ziel>`

## Beispiele

- `cp bsp bspkopie` – kopiert die Datei 'bsp' nach 'bspkopie' (im aktuellen Verzeichnis)
- `cp bsp test/` – kopiert die Datei 'bsp' in das Verzeichnis 'test'
- `cp -v bsp test/` – ... mit Ausgabe der einzelnen Kopieraktionen
- `cp -r test/ test2` – erstellt eine Kopie des Verzeichnisses 'test' mit dem Namen 'test2'
- `cp -r /verz .` – erstellt eine Kopie des Verzeichnisses '/verz' im aktuellen Verzeichnis



# Elementare Befehle

mkdir, rmdir – Verzeichnisse erstellen und entfernen

## mkdir

`mkdir foo` legt ein Verzeichnis 'foo' im aktuellen Verzeichnis an

## rmdir

`rmdir foo` löscht das Verzeichnis 'foo' aus dem aktuellen Verzeichnis ('foo' muss leer sein)

# Elementare Befehle

rm – Löschen

rm

rm löscht Dateien und Verzeichnisse

## Beispiele

- rm foo.pdf – löscht die Datei 'foo.pdf'
- rm -r Mails/ – löscht das Verzeichnis 'Mails' und alle darin enthaltenen Dateien und Unterverzeichnisse
- rm -rf wichtig/ – löscht das Verzeichnis 'wichtig' mit allen darin enthaltenen Dateien und Unterverzeichnissen, ohne nachzufragen – auch falls diese schreibgeschützt sind!

## Achtung!

rm löscht **ohne** Nachfrage und **ohne** Umweg über den Papierkorb!

# Elementare Befehle

## Anzeige von Textdateien

Zum Anzeigen von Textdateien gibt es den Befehl `cat`.

### Typische Verwendung

```
cat <Datei>
```

```
$ cat elementare-befehle.tex
\begin{frame}
\frametitle{man-pages -- das Hilfesystem unter Unix}
...
```

# Elementare Befehle

## Anzeige von Textdateien (2)

Hilfe, so schnell kann ich nicht lesen!

### Wie kann ich die Anzeige verlangsamen?

cat gibt eingelesene Datei komplett aus, egal wie groß diese ist.  
Seitenweise Anzeige: less.

### Typische Verwendung

```
less <Datei>
```

### Achtung!

- cat und less können nur Textdateien sinnvoll anzeigen.
- Falls nach der Ausgabe einer Binärdatei nur noch seltsame Zeichen dargestellt werden, hilft der Befehl reset.

# Wildcards

```
$ ls  
vorkurs2014.aux vorkurs2014.log vorkurs2014.nav  
vorkurs2014.pdf vorkurs2014.tex vorkurs2014.toc  
vorkurs2015.aux vorkurs2015.log vorkurs2015.nav  
vorkurs2015.pdf vorkurs2015.tex vorkurs2015.toc
```

Wie werde ich nur die ganzen Dateien vom letzten Jahr los?

```
$ rm vorkurs2014.aux vorkurs2014.log vorkurs2014.nav ...
```

Geht das nicht einfacher?!

# Wildcards

Aber natürlich.

## Platzhalter

Die *bash* erlaubt den Einsatz von Platzhalterzeichen („Wildcards“).

- \* steht für beliebig viele (oder auch keine) Zeichen
- ? steht für genau ein Zeichen

Zurück zum Beispiel:

```
$ rm vorkurs2014*
```

vorkurs2014\* steht demnach für alle Dateinamen, die mit vorkurs2014 beginnen:

```
vorkurs2014* ↪ vorkurs2014.aux vorkurs2014.log ...
```

## Platzhalter II

Es geht auch noch etwas komplizierter:

- [123] steht für genau eines der Zeichen zwischen den eckigen Klammern: 1 2 3
- [!123] steht für ein Zeichen, das nicht zwischen den Klammern steht: z.B. a 4 J \_
- [a-d] steht für ein Zeichen aus dem angegebenen Bereich: a b c d
- {1,2,abc} steht der Reihe nach für *alle* der angegebenen Strings (unabhängig davon, ob eine Datei mit dem Namen existiert)

# Wildcards

## Beispiele

```
$ ls  
hand sand band  
  
$ echo [hbr]and  
hand band
```

```
$ wget http://www.example.net/folien{0,1,2,3,4}.pdf
```

Lädt die Dateien folien0.pdf, folien1.pdf, ... vom Server herunter

```
$ pdftk folien*.pdf cat output allefolien.pdf
```

... und baut die heruntergeladenen Dateien folien0.pdf, folien1.pdf, folien2.pdf, ... zu einer großen PDF-Datei zusammen.



# Wildcards

## Hinweis

Der \*-Platzhalter bezieht sich nur auf nicht-versteckte Dateien!

```
$ ls -a
.      ..      .bash_history  a.txt  mein_bild.jpg
$ rm *
$ ls -a
.      ..      .bash_history
```

## Achtung!

`rm .*` würde `.` theoretisch zu `..` expandieren!  
(die meisten `rm`-Versionen überprüfen das allerdings intern)

# Drucken im CIP-Pool

## Allgemeines

`lpr`

`lpr` druckt ein PDF- bzw. PS-Dokument aus.

## Beispiel

```
lpr -Pps1bcipd foo.pdf - druckt die Datei 'foo.pdf' auf dem  
Drucker 'ps1bcipd' aus
```

# Drucken im CIP-Pool

## Druckernamen

### ⟨Drucker⟩ – Druckernamen

ps⟨Stockwerk⟩⟨Buchstabe⟩cip⟨Suffix⟩⟨Doppelseitig⟩

⟨Stockwerk⟩	in welchem der Drucker steht
⟨Buchstabe⟩	Unterscheidung der einzelnen Drucker
⟨Doppelseitig⟩	d – Duplex lange Seite t – Duplex kurze Seite weglassen – kein Duplex

Die Namen der Drucker sind auch am Gerät abzulesen.

### Drucker ps2ccip

- Farbig drucken (teurer!); schwarz-weiß mit Suffix bw erzwingen
- Scannen (siehe Anleitung, die über dem Drucker an der Wand hängt)

# Drucken im CIP-Pool

## Druckerwarteschlange

lpq

lpq zeigt die aktuelle Druckerwarteschlange an.

```
$ lpq -P ps2ccipbw
Printer: ps2ccipbw@faiu06 (dest ps2ccipbw@faiu0print.informatik.uni-erlangen.de)
Queue: no printable jobs in queue
Server: no server active
Status: job 'he29heri@faiu06+782' saved at 14:54:35.244
Rank  Owner/ID                Pr/Class Job Files          Size Time
done  he29heri@faiu06+782      A    782 (STDIN)             381966 14:54:35
Printer: ps2ccipbw@faiu0print (dest ps2ccipprivate@localhost)
Queue: no printable jobs in queue
Server: no server active
Status: job 'he29heri@faiu06+782' saved at 14:57:34.635
Rank  Owner/ID                Pr/Class Job Files          Size Time
done  he29heri@faiu06+782      A    782 (STDIN)             381966 14:54:35
Printer: ps2ccipprivate@faiu0print
Queue: no printable jobs in queue
[...]
```

# Drucken im CIP-Pool

## Druckaufträge löschen

### lprm

lprm löscht Druckaufträge aus der Druckerwarteschlange.

### Beispiele

- |                  |   |  |
|------------------|---|--|
| lprm             | – | löscht den letzten Druckauftrag  |
| lprm -P<Drucker> | – | löscht alle eigenen Druckaufträge aus der Druckerwarteschlange von <Drucker> |
| lprm -a          | – | löscht alle eigenen Druckaufträge aus allen Druckerwarteschlangen            |

# Drucken im CIP-Pool

## Tipps

### pr\_acct

pr\_acct zeigt das aktuelle Druckguthaben an.

Das Druckerkonto kann im Sekretariat (02.155) durch Bareinzahlung aufgeladen werden.

### PDF manipulieren

- |         |  |
|---------|--|
| pdfnup  | PDFs anders anordnen (z. B. 4 Folien auf eine Seite)                 |
| pdftk   | kann PDFs allgemein manipulieren (z. B. mehrere PDFs zusammenführen) |
| pdftops | wandelt PDF in PS um   |

42

## Referenzen

- <https://en.flossmanuals.net/command-line/>
- [https://fsi.cs.fau.de/repo\\_public/vorkurs/](https://fsi.cs.fau.de/repo_public/vorkurs/)