Linux-Kurs – Teil 1

FSI Informatik

FAU Erlangen-Nürnberg

17. Oktober 2023

Allgemeines

Mittwoch 19.04.2023

Zeit	Raum	Inhalt
12:00–14:00	H6	Vorlesung (Teil 1)
14:00–16:00	CIP2	Übung (Teil 1)

Donnerstag 20.04.2023

Zeit	Raum	Inhalt
12:00–14:00	H9	Vorlesung (Teil 2)
14:00–16:00	CIP2	Übung (Teil 2)

Linux ist eine grundlegende Komponente von freien und offenen Betriebsystemen, neben anderen wie Fensterverwaltung oder Arbeitsprogrammen.

Linux ist eine grundlegende Komponente von freien und offenen Betriebsystemen, neben anderen wie Fensterverwaltung oder Arbeitsprogrammen.
Das gesamte Paket wird "Distribution" genannt (bspw. "Debian", "Ubuntu", "Fedora", "OpenSuSe", ...).

Linux ist eine grundlegende Komponente von freien und offenen Betriebsystemen, neben anderen wie Fensterverwaltung oder Arbeitsprogrammen.
Das gesamte Paket wird "Distribution" genannt (bspw. "Debian", "Ubuntu", "Fedora", "OpenSuSe", ...).
Linux teilt ein technisches Erbe mit vielen anderen Betriebsystemen aus der UNIX-Familie wie macOS oder Android, und viel Wissen lässt sich übertragen.

CIP-Pools im Blauen Hochhaus:

- Linux-Arbeitsrechner
- Drucker
- Farbdrucker+Scanner im CIP 2

Allgemeines

Etiquette im CIP

- CIPs sind Arbeitsräume – nicht zu laut sein!

Allgemeines

Etiquette im CIP

- CIPs sind Arbeitsräume – nicht zu laut sein!
- Achtsam mit der Hardware (und Software) umgehen.

Allgemeines

Etiquette im CIP

- CIPs sind Arbeitsräume – nicht zu laut sein!
- Achtsam mit der Hardware (und Software) umgehen.
- Beim Verlassen des Arbeitsplatzes Bildschirm sperren.

Allgemeines

Etiquette im CIP

- CIPs sind Arbeitsräume – nicht zu laut sein!
- Achtsam mit der Hardware (und Software) umgehen.
- Beim Verlassen des Arbeitsplatzes Bildschirm sperren.
- Bei Problemen keine Angst haben sich umzufragen – Studenten helfen einander gerne.

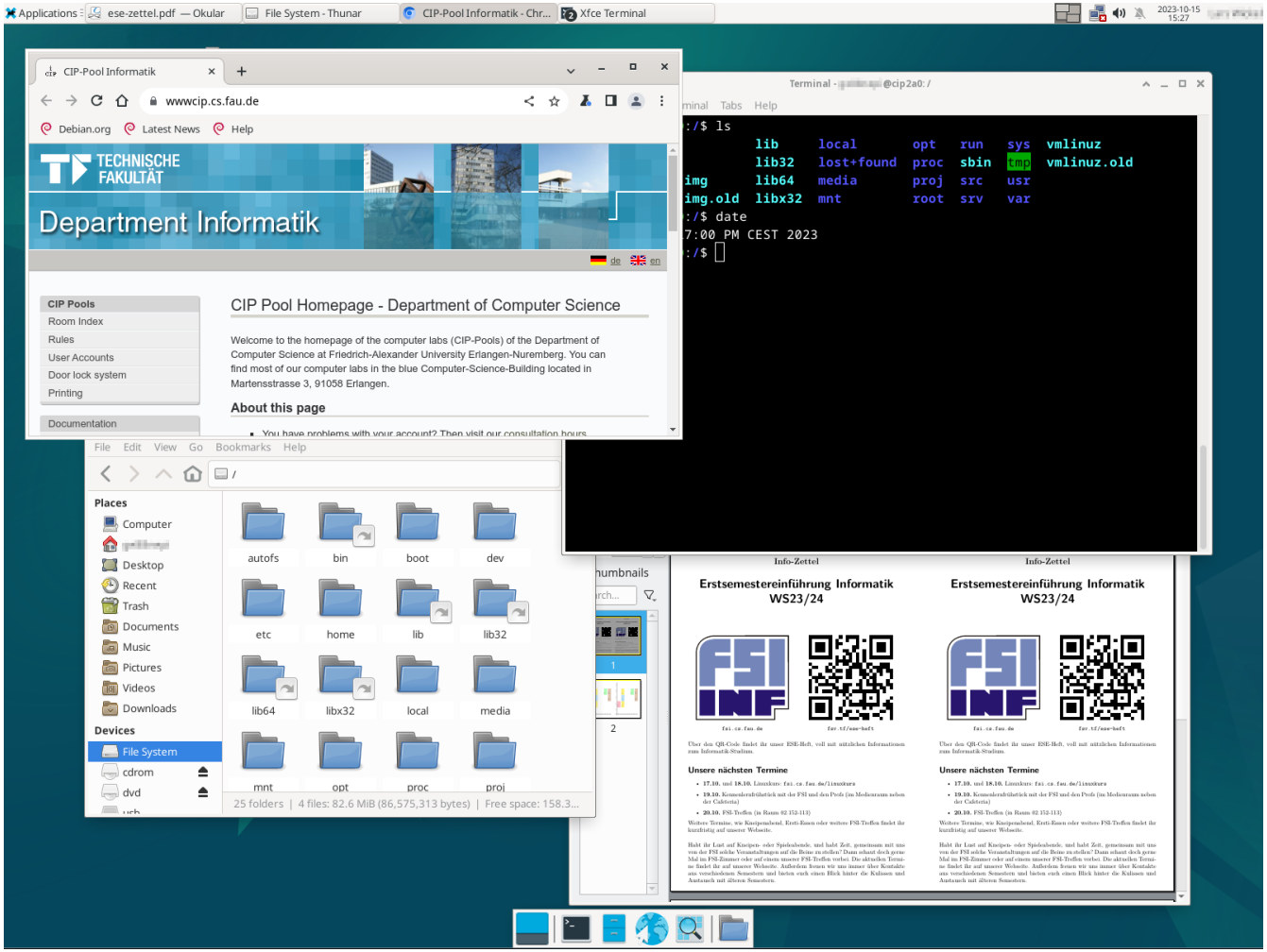
Allgemeines

Etiquette im CIP

- CIPs sind Arbeitsräume – nicht zu laut sein!
- Achtsam mit der Hardware (und Software) umgehen.
- Beim Verlassen des Arbeitsplatzes Bildschirm sperren.
- Bei Problemen keine Angst haben sich umzufragen – Studenten helfen einander gerne.
- Essen und Trinken verboten! (Loginentzug droht)

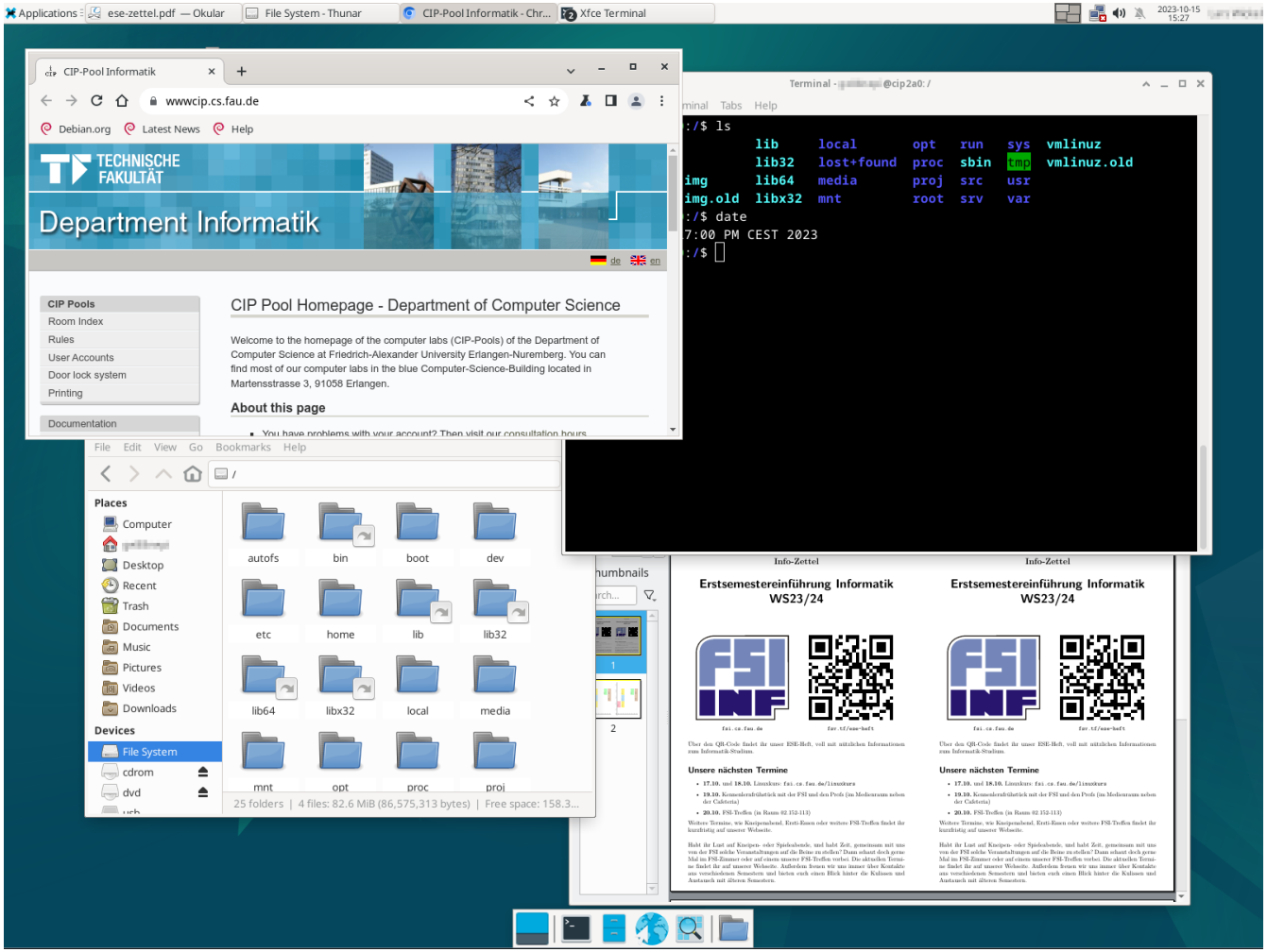
Allgemeines

Graphische Benutzeroberfläche, nichts neues. . .



Allgemeines

Graphische Benutzeroberfläche, nichts neues. . .

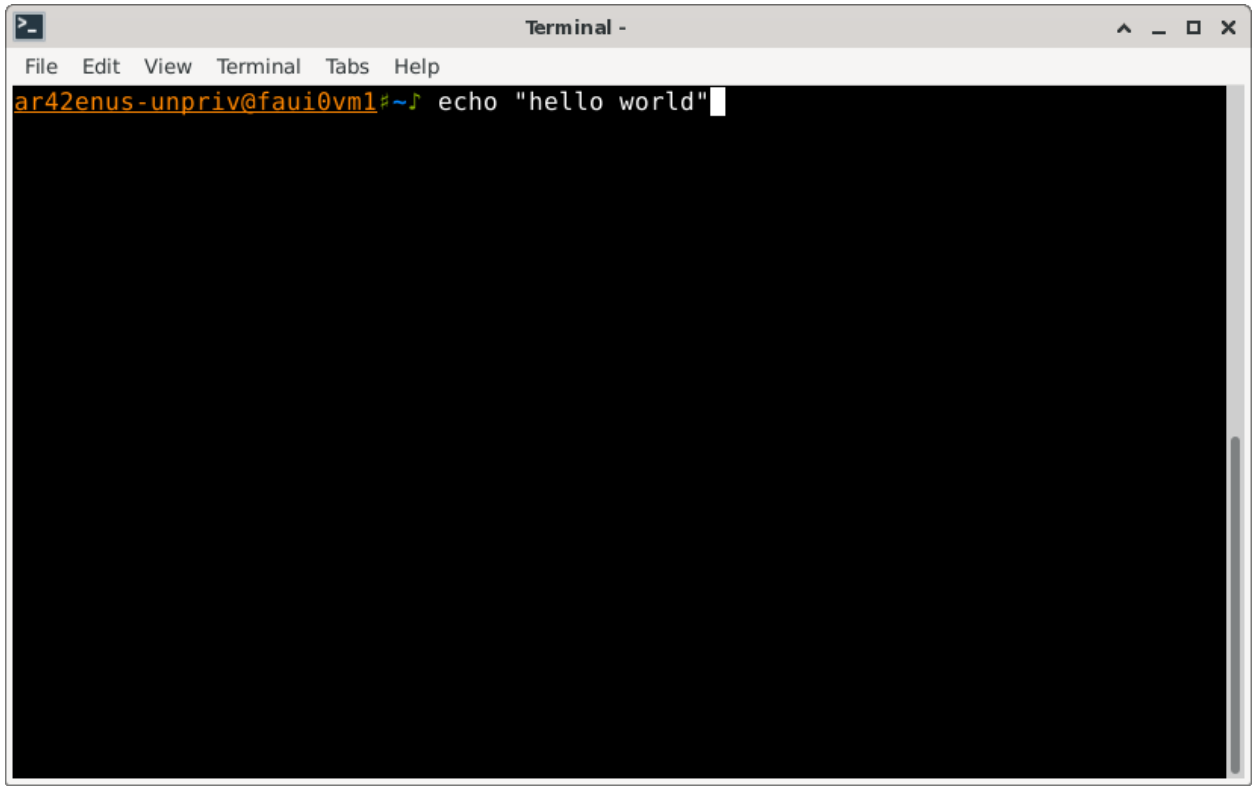


. . . bis auf das kleine schwarze Fenster: Ein Terminal.

Befehlszeile

Befehlszeile

Befehlszeile – Warum?



Selbst nach 60 Jahren immer noch beliebt und viel Benutzt. Was macht es so interessant?

Befehlszeile

Beispiele für nützliche ~~Zaubersprüche~~ Befehle

Herunterladen eines Videos

```
$ yt-dlp 'https://youtu.be/dQw4w9WgXcQ'
```

Umwandelen von Dateiformaten

```
$ pandoc 'https://example.org/' -o example.docx
```

Extrahieren des Audio-Kanals aus einem Video

```
$ ffmpeg -i video.avi sound.mp3
```

Suchen nach Textmustern in Dateien

```
$ grep -w 'Li.*x' /usr/share/dict/words
```

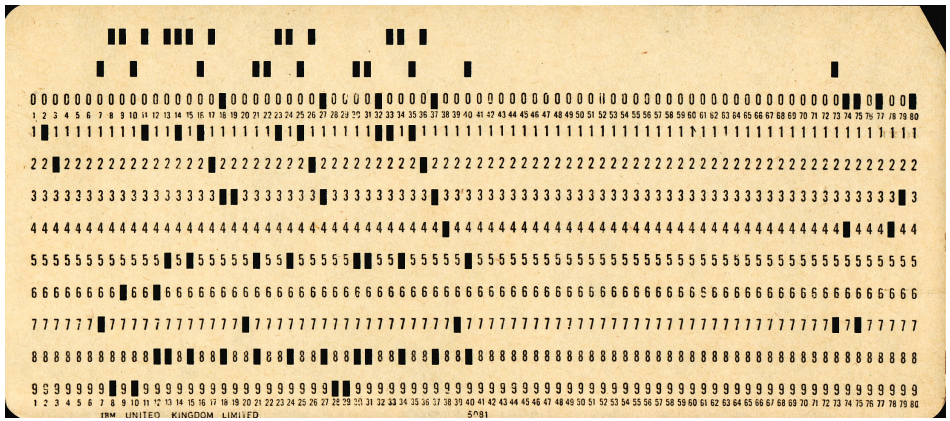
Installieren von Anwendungen

```
$ apt install blender vlc
```

Terminal

Geschiche im Kontext

- Frühe Systeme Setzen auf Lochkarten



Geschiche im Kontext

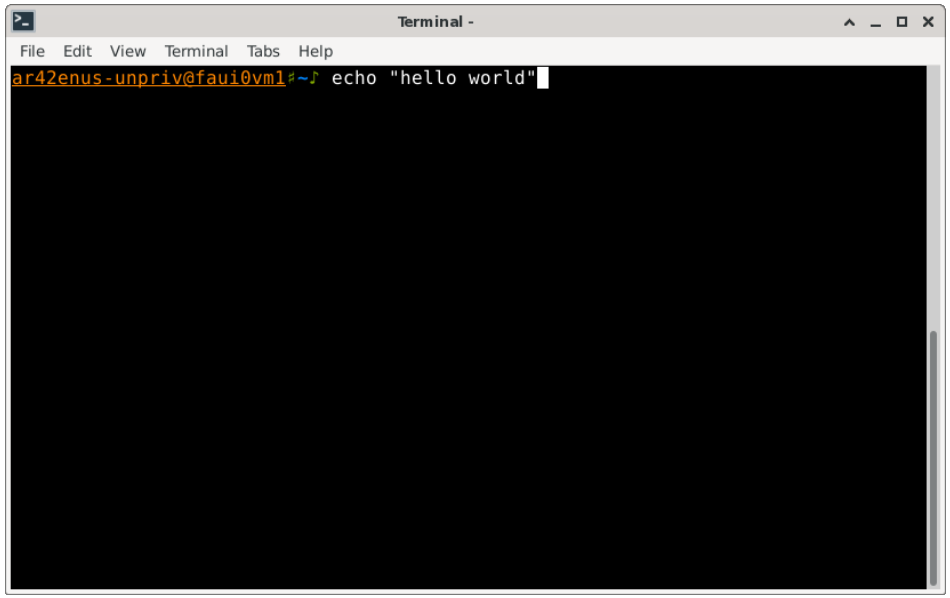
- Frühe Systeme Setzen auf Lochkarten
- Interaktiver gebraucht mit Fernschreibern ermöglicht

Geschiche im Kontext

- Frühe Systeme Setzen auf Lochkarten
- Interaktiver gebraucht mit Fernschreibern ermöglicht
- Video-Terminals ermöglichen mehr Funktionalitäten

Geschiche im Kontext

- Frühe Systeme Setzen auf Lochkarten
- Interaktiver gebraucht mit Fernschreibern ermöglicht
- Video-Terminals ermöglichen mehr Funktionalitäten
- Bis auch diese ersetzt aber emuliert werden, samt aller historischen Eigenartigkeiten



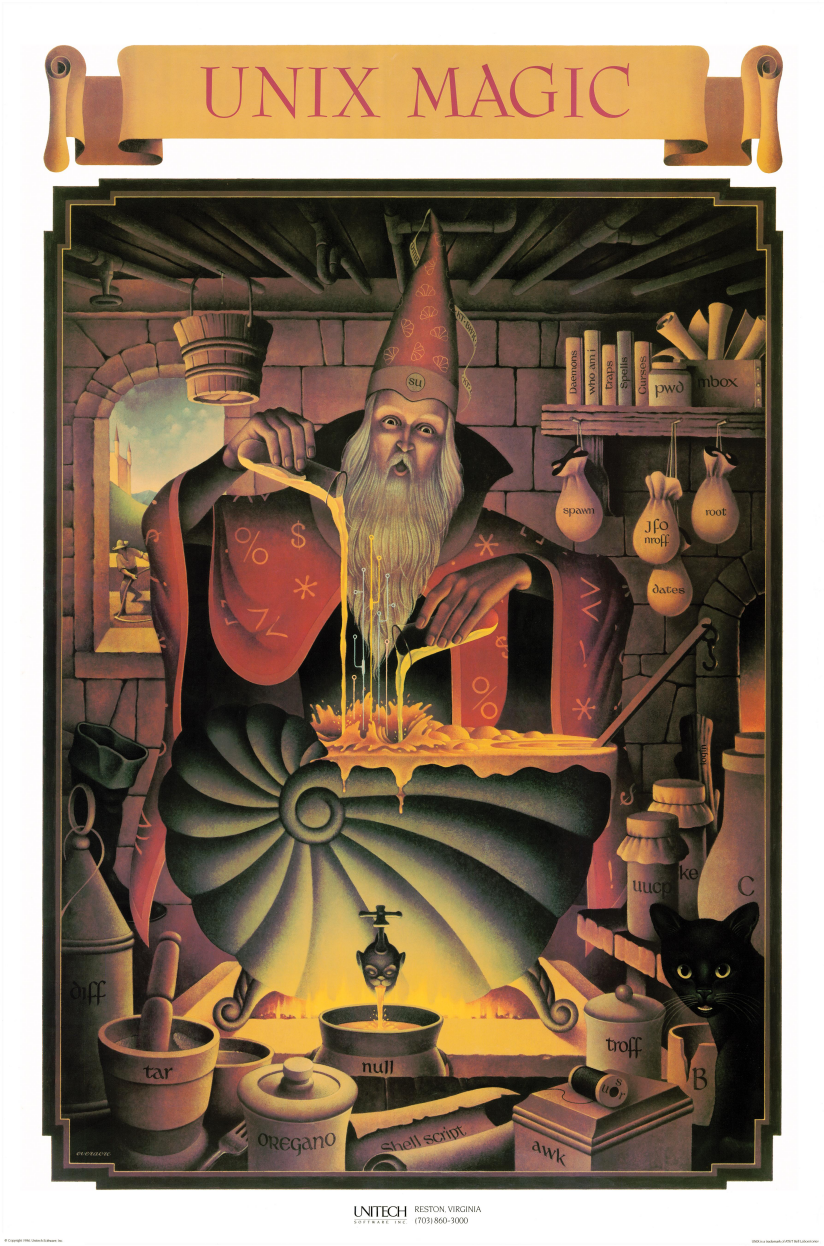
Die Shell (Außenhaut, Randzone, Ummantelung)

- Durch all diese Entwicklungen bleibt die Shell konstant.
- Es verarbeitet Eingaben und verwaltet Programme.
- Wir schauen das alles nun in der Praxis an. . .

Die Shell (Außenhaut, Randzone, Ummantelung)

- Durch all diese Entwicklungen bleibt die Shell konstant.
- Es verarbeitet Eingaben und verwaltet Programme.
- Wir schauen das alles nun in der Praxis an. . .

Die Shell (Außenhaut, Randzone, Ummantelung)



- Durch all diese Entwicklungen bleibt die Shell konstant.
- Es verarbeitet Eingaben und verwaltet Programme.
- Wir schauen das alles nun in der Praxis an. . .

Die nun folgende, Rosa hervorgehobenen Folien wurden alle während des Vortrags übersprungen, und mit einem praktischem Teil ersetzt. Dieser Abschnitt kann aber dennoch im Selbststudium als Referenz genutzt werden.

Befehlsaufbau

Befehlsaufbau

Allgemein

Im Terminal kann man jetzt Befehle eingeben:

```
$ echo
```

Befehlsaufbau

Allgemein

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
```

Also einmal mit zwei Wörtern:

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

Befehlsaufbau

Mehrere Parameter

Also einmal mit zwei Wörtern:

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

. . . und noch ein paar Leerzeichen mehr:

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

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[1] aufheben – der Text, der in Anführungszeichen steht, wird als ein einziger langer Parameter interpretiert.

Lösung:

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

[1] und anderen Sonderzeichen

Befehlsaufbau

Optionen

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 $ _
```

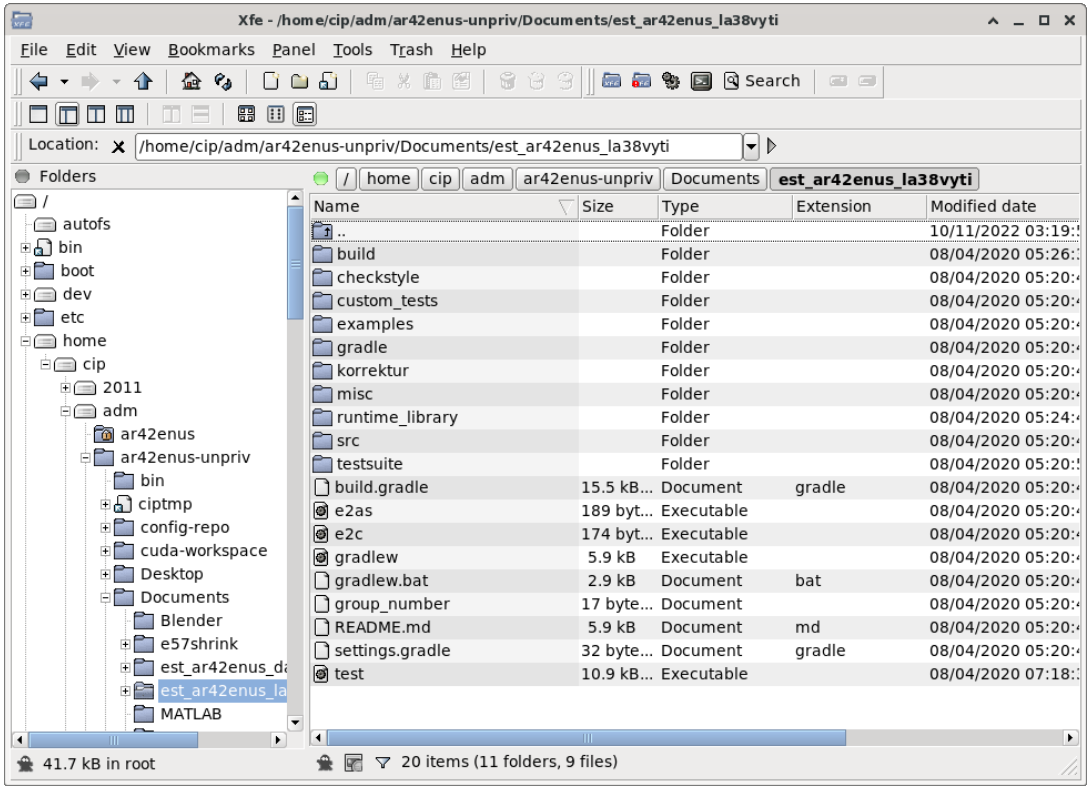
Herumklettern im Dateisystembaum

Herumklettern im Dateisystembaum

Hilfe! Wo ist der Explorer?

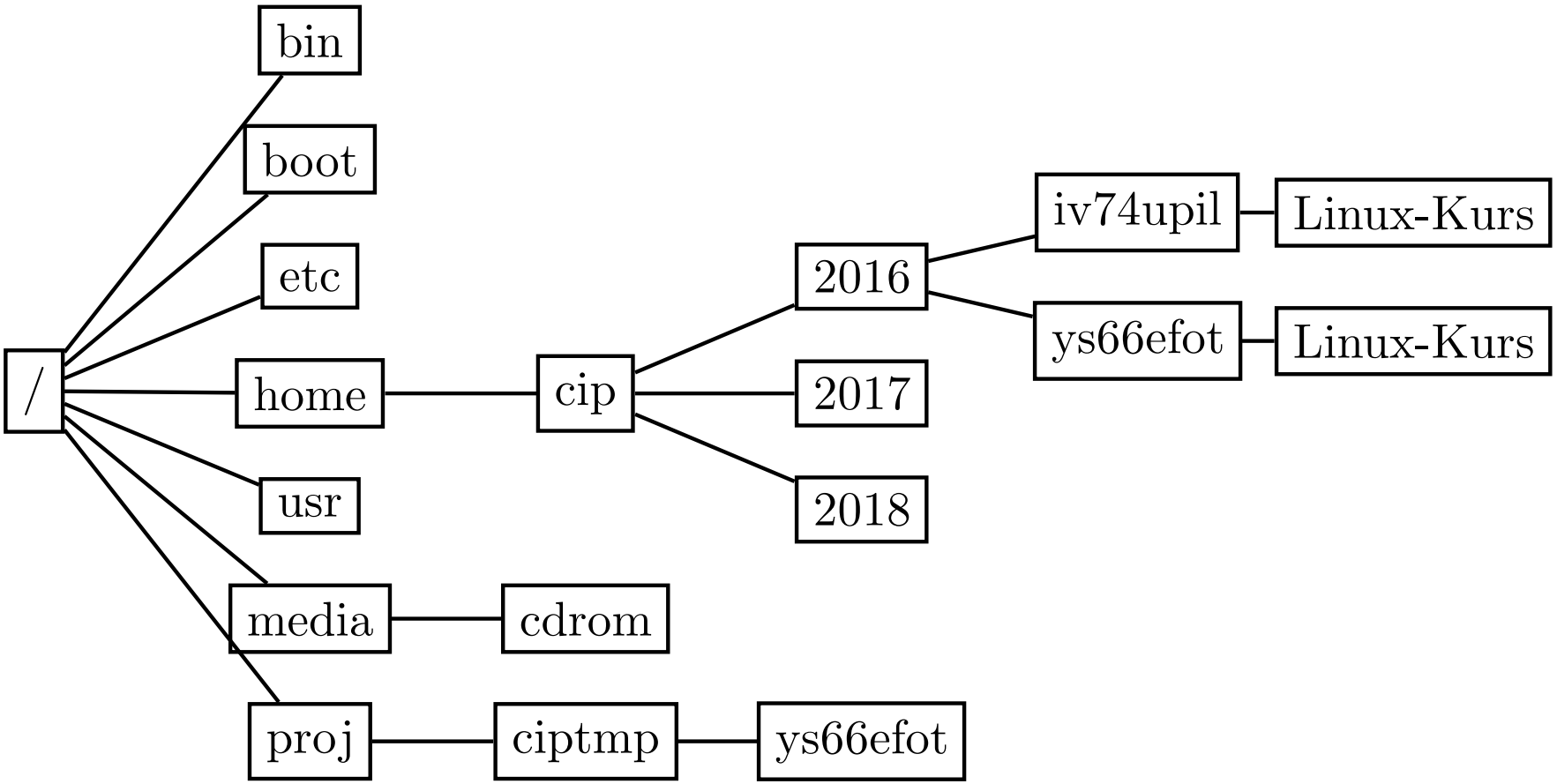
Noch schnell: graphische Dateibrowser für den Notfall:

- Nautilus
- Dolphin
- Thunar
- Xfe
- ...



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!

```
C:\Users\klaus c:\users\KLaUs
/home/klaus
```

Herumklettern im Dateisystembaum

`cip-mountusb` – USB-Sticks einhängen

USB im CIP

`cip-mountusb`	hängt den USB-Stick unter `/media/usb` ein
`cip-umountusb`	hängt den USB-Stick wieder aus

Anmerkungen

- Vor dem Abziehen des Sticks unmounten $\rightarrow$ sonst Datenverlust!
- FAT32 und NTFS wird unterstützt

Herumklettern im Dateisystembaum

Wo zum Teufel sind wir überhaupt?

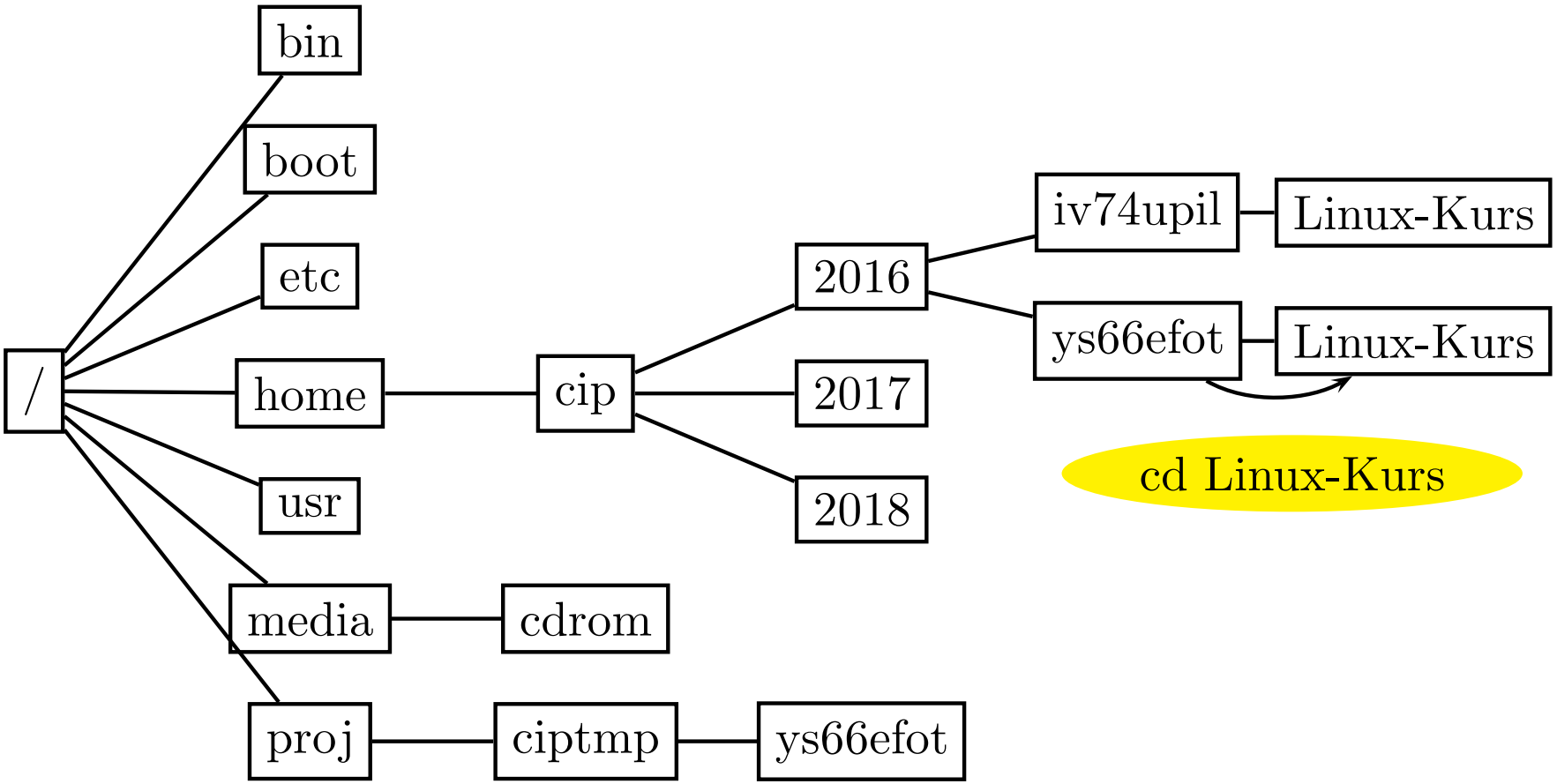
`pwd`

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

```
$ pwd
/home/cip/2016/ys66efot
```

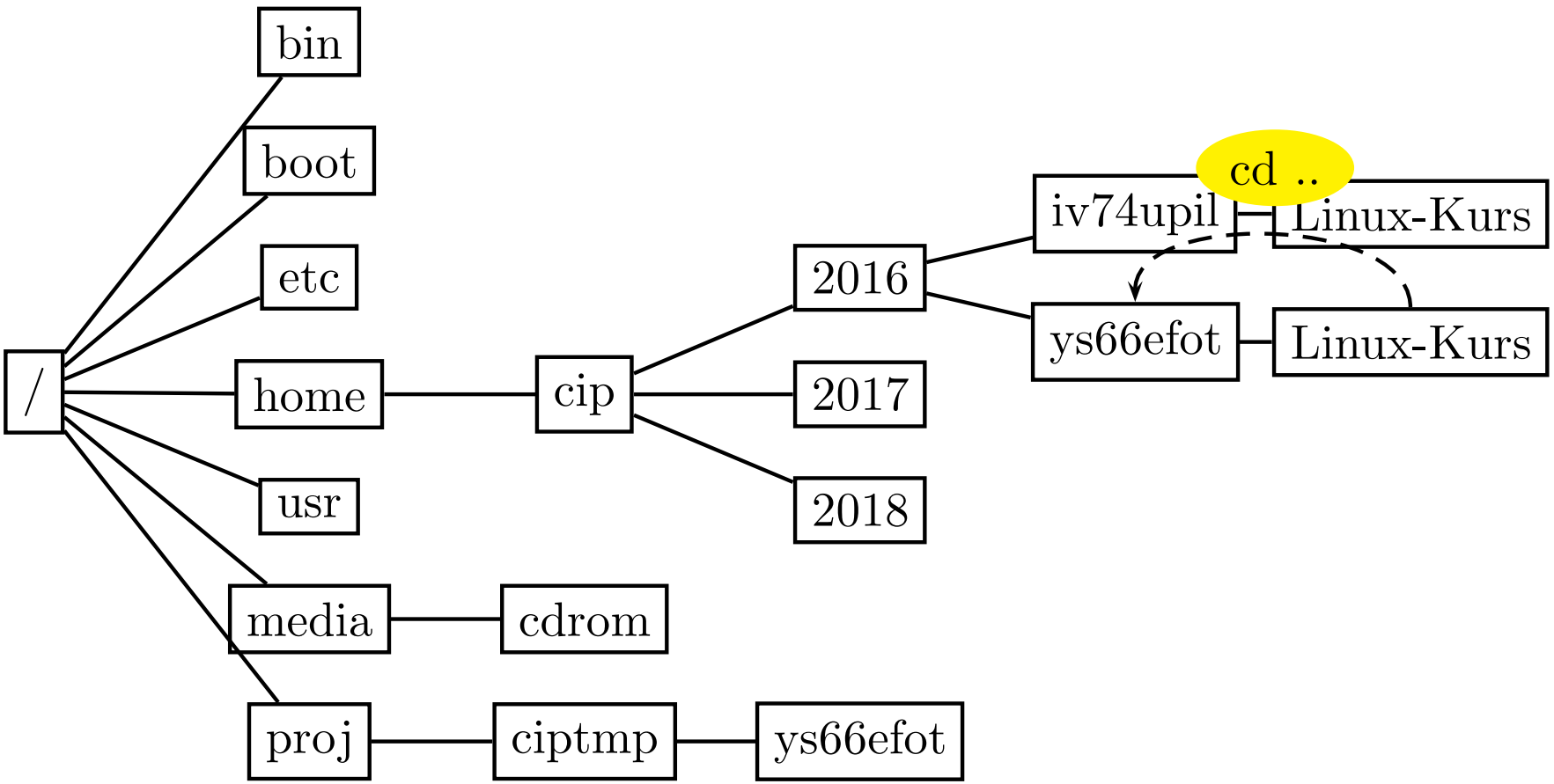
Herumklettern im Dateisystembaum

Verzeichniswechsel



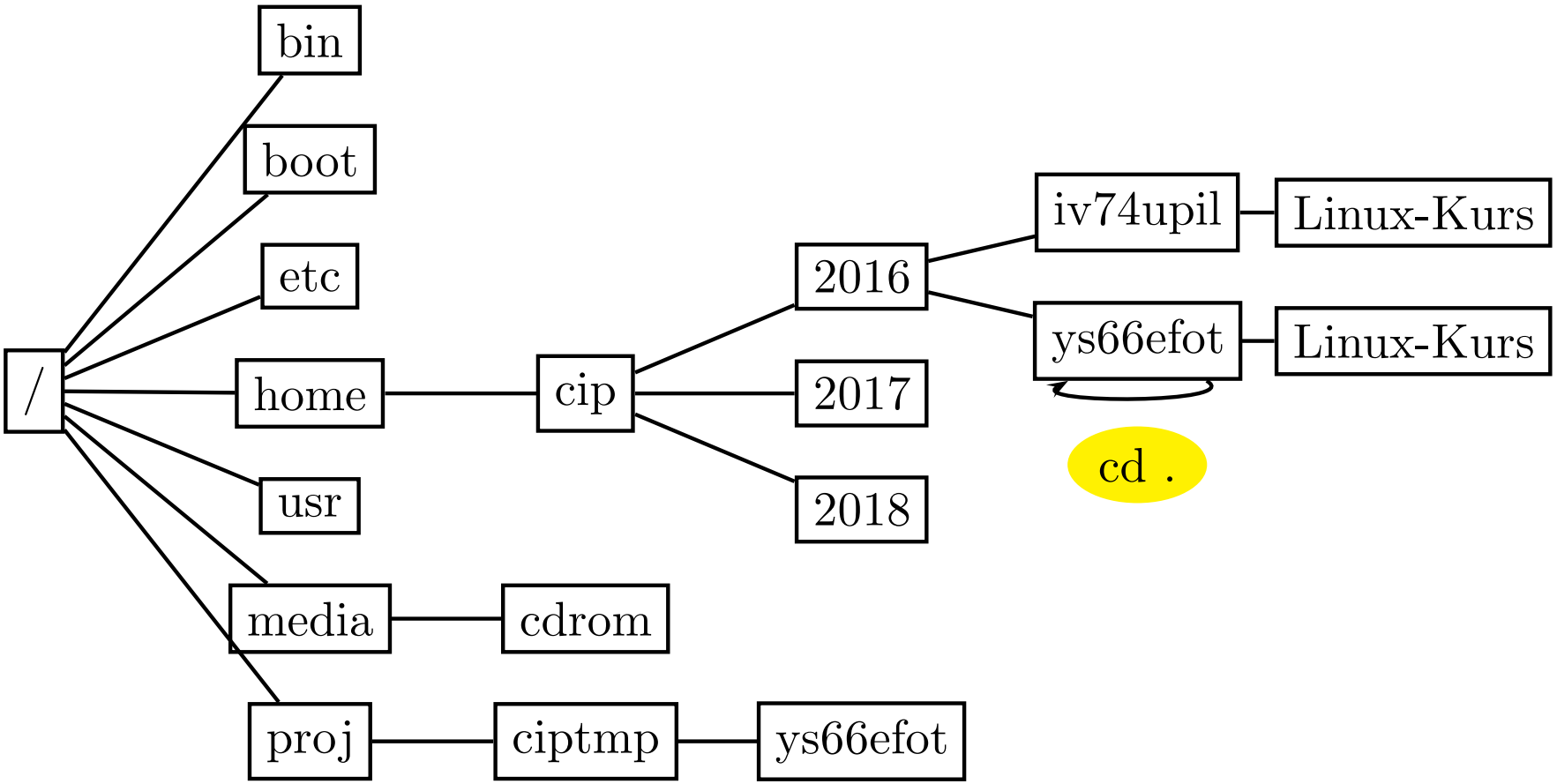
Herumklettern im Dateisystembaum

Verzeichniswechsel ins übergeordnete Verzeichnis



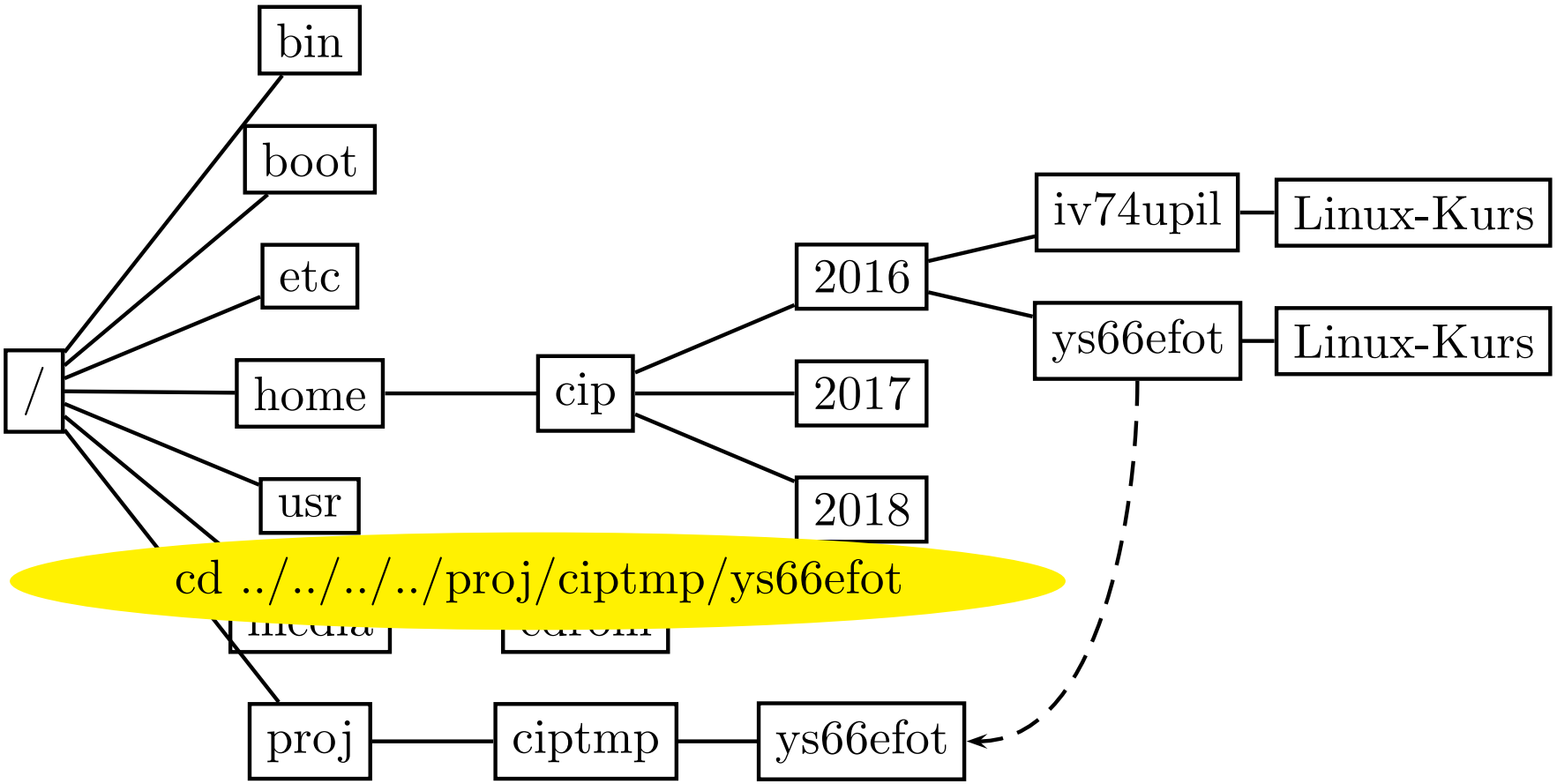
Herumklettern im Dateisystembaum

„Verzeichniswechsel“



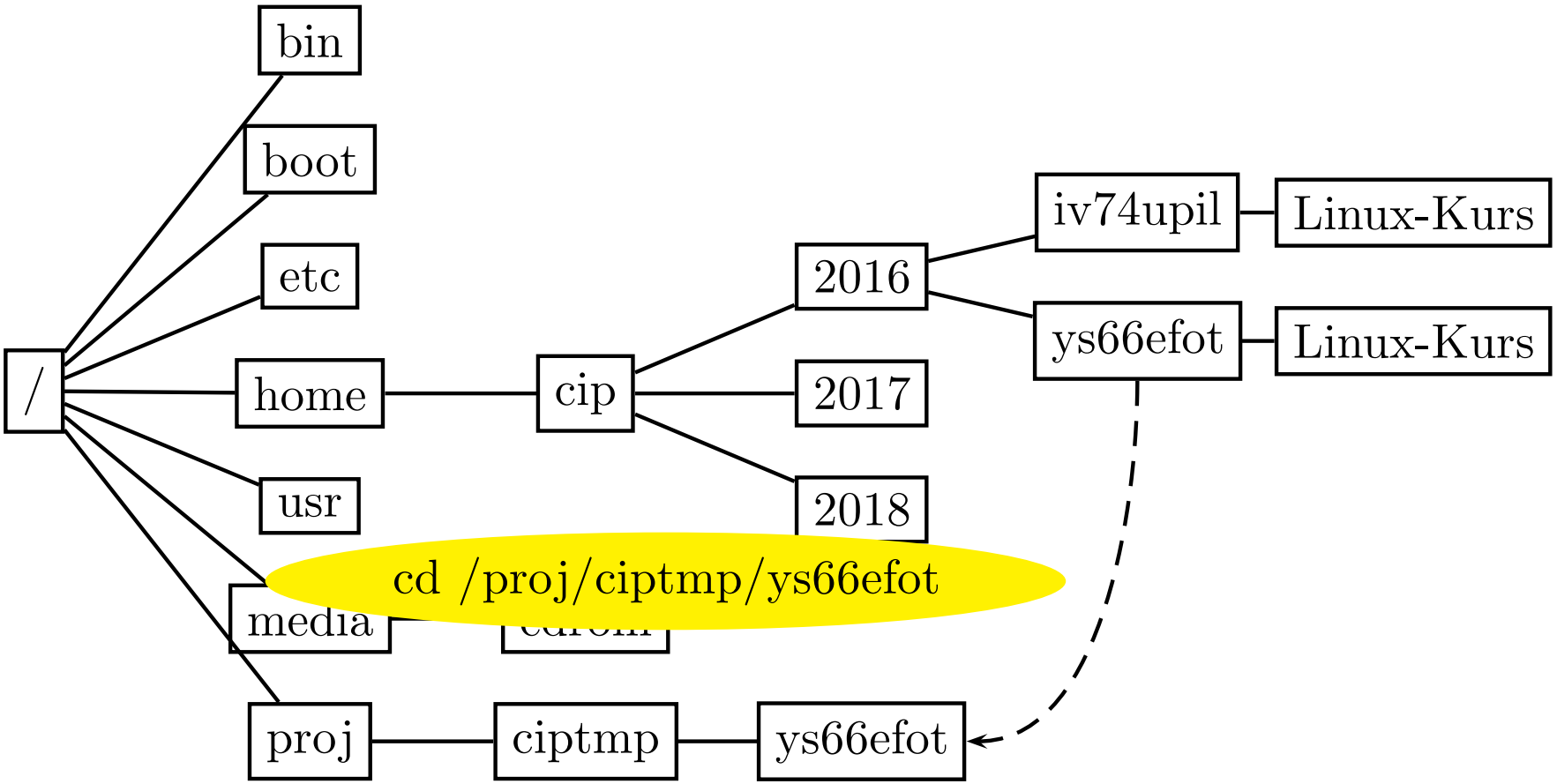
Herumklettern im Dateisystembaum

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/2022/<userlogin>`):
  - Es steht nur begrenzter Speicherplatz zur Verfügung
  - Dort liegen Konfigurationen und Nutzdaten
  - Der Inhalt wird täglich gesichert und ist zentral gespeichert, also auf allen Rechnern gleich
  - Kurzschreibweise fürs *Home*-Verzeichnis: $\sim$ (*Tilde-Zeichen*)

- Mehr Speicherplatz (8 GB) 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!)

Der Befehl `cip-quota` zeigt, wie viel Speicherplatz zur Verfügung steht.

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

Herumklettern im Dateisystembaum

Speicherplatzverbrauch – interaktiv per Konsole mit `ncdu`

```
$ ncdu /etc
```

```
ncdu 1.15.1 ~ Use the arrow keys to navigate, press ? for help
--- /etc ------------------------------------------------------------------------
    6.3 MiB [##########] /libreoffice
    1.4 MiB [##        ] /X11
    1.0 MiB [#         ] /scite
    1.0 MiB [#         ] /xdg
  800.0 KiB [#         ] /ssh
. 760.0 KiB [#         ] /ssl
  712.0 KiB [#         ] /mono
  588.0 KiB [          ] /direct
  496.0 KiB [          ]  avrdude.conf
  496.0 KiB [          ] /java-17-openjdk
  496.0 KiB [          ] /java-11-openjdk
  496.0 KiB [          ] /asciidoc
  484.0 KiB [          ] /timidity
  472.0 KiB [          ]  ld.so.cache
  380.0 KiB [          ] /xpra
  340.0 KiB [          ] /sane.d
  312.0 KiB [          ] /joe
  288.0 KiB [          ] /init.d
  280.0 KiB [          ] /ImageMagick-6
  260.0 KiB [          ] /nagios-plugins
  252.0 KiB [          ] /systemd
  216.0 KiB [          ] /alternatives
 Total disk usage:  25.8 MiB  Apparent size:  20.0 MiB  Items: 4320
```

Inhalte aufzeigen

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 Verzeichnisauflistung (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 *versteckt* interpretiert und nicht angezeigt, z. B.:
  - . ist immer das aktuelle Verzeichnis
  - .. ist immer das übergeordnete Verzeichnis
  - .bash_history enthält z. B. Befehle, die früher eingegeben wurden

Fahrt aufnehmen

Fahrt aufnehmen

Tab-Vervollständigung

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

- Namen von Befehlen
- Datei- und Verzeichnisnamen

```
$ ls
Desktop   folien_linuxkurs_tag1.pdf

$ file f<TAB>
$ file folien_linuxkurs_tag1.pdf
folien_linuxkurs_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
aufgaben_linuxkurs_tag1.pdf  folien_linuxkurs_tag1.pdf
aufgaben_linuxkurs_tag2.pdf  folien_linuxkurs_tag2.pdf

$ file f<TAB>
$ file folien_linuxkurs_tag<TAB><TAB>
folien_linuxkurs_tag1.pdf  folien_linuxkurs_tag2.pdf
$ file folien_linuxkurs_tag2<TAB>
$ file folien_linuxkurs_tag2.pdf
folien_linuxkurs_tag2.pdf: PDF document, version 1.4
```

Fahrt aufnehmen

Suche in der Befehlshistory

- Mit Cursortasten hoch/runter durch letzte Befehle bewegen

Fahrt aufnehmen

Suche in der Befehlshistory

- Mit Cursortasten hoch/runter durch letzte Befehle bewegen
- `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.
- ... sehr praktisch für lange komplizierte Zeilen!

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.

Viele Terminals unterstützen auch `Ctrl+Shift+C` und `Ctrl+Shift+V`.

Elementare Befehle

Elementare Befehle

manpages – 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
```

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 (terminalbasierte) Programme wie `less` lassen sich so bedienen!

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

`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

`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

mv, cp, rm – interaktive Nachfrage

Achtung!

mv und cp überschreiben Dateien mit dem gleichen Namen!

-i

mv -i, cp -i	–	fragt jedes Mal interaktiv nach, wenn eine Datei überschrieben werden würde
rm -i	–	fragt bei jeder Datei und jedem Verzeichnis nach, ob sie gelöscht werden sollen

Zur Ausgabe von Textdateien gibt es den Befehl `cat`.

Typische Verwendung

cat <Datei>

```
$ cat elementare-befehle.tex
\begin{frame}
\frametitle{manpages -- 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

Wildcards

```
$ ls
linuxkurs2022.aux linuxkurs2022.log linuxkurs2022.nav
linuxkurs2022.pdf linuxkurs2022.tex linuxkurs2022.toc
linuxkurs2023.aux linuxkurs2023.log linuxkurs2023.nav
linuxkurs2023.pdf linuxkurs2023.tex linuxkurs2023.toc
```

Wildcards

```
$ ls
linuxkurs2022.aux linuxkurs2022.log linuxkurs2022.nav
linuxkurs2022.pdf linuxkurs2022.tex linuxkurs2022.toc
linuxkurs2023.aux linuxkurs2023.log linuxkurs2023.nav
linuxkurs2023.pdf linuxkurs2023.tex linuxkurs2023.toc
```

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

```
$ rm linuxkurs2022.aux linuxkurs2022.log linuxkurs2022.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

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 linuxkurs2022*
```

`linuxkurs2022*` steht demnach für alle Dateinamen, die mit `linuxkurs2022` beginnen:
`linuxkurs2022*` $\rightsquigarrow$ `linuxkurs2022.aux linuxkurs2022.log ...`

Wildcards

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
$ ls [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)

Referenzen

- `https://fsi.cs.fau.de/linuxkurs`
- intro(1)
- `https://explainshell.com/`