Linux Basics

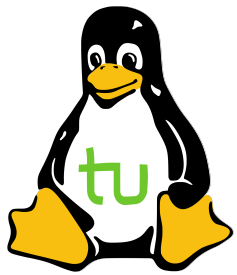
— Das Terminal —

Jan-Marius Lenk, Christoph Parnitzke, Yannick Bungers,
Alexander Becker

Free and Open Source Software AG
Fakultät für Informatik
20. April 2017

Inhaltsverzeichnis

Unix Philosophie



- Philosophie besteht aus drei Punkten:
  - 1. Schreibe Programme, die nur eine Sache tun und dies erfolgreich
  - 2. Schreibe Programme, die Kolaboration ermöglichen
  - 3. Schreibe Programme, die mit Text arbeiten, denn dies ist universell

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

list
Ein kleines Licht in der Dunkelheit

- Listet alle Ordner und Dateien in Verzeichnis auf
- Aktuelles Verzeichnis ist dabei Default
- Mögliche Argumente (Default: -1):
  - `-a` (all) inkl. versteckter Verzeichnisse
  - `-l` (long listing format) inkl. Rechte, Besitzer, Größe, etc.
  - `-h` (human readable) Größen der Dateien leserlicher
  - `-r` (reversive) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (recursive) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit `~`)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit `~`)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit `~`)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit `~`)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit ~)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit ~)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

change directory
Die kleine Form der Teleportation

`cd` ermöglicht das Navigieren durch das Dateisystem

- Notationen:
  - Wechsel in Parent-Directory mit `..`
  - Wechsel in Home-Directory mit `cd` (ohne Pfad)
  - Wechsel in Verzeichnis (Bsp.: Dokumente (bzw. Documents))
    - `cd /home/[username]/Dokumente`
    - `cd ~/Dokumente` (Abkürzen von Home-Directory mit ~)
    - `cd ~[username]/Dokumente`
    - `cd Dokumente` (relativ vom aktuellen Verzeichnis)
- Funfact: `cd .` (wechselt in das aktuelle Verzeichnis)
  - es tut nichts, aber dafür sehr gut

find

Mit `find` lassen sich Dateien einfach im aktuellen Ordner und in allen Unterordnern suchen

- `find -name [path]`
  - Bsp.: `find -name output.txt`

make directory
Verzeichnisse erschaffen

`mkdir` erstellt Ordner auf dem Rechner

- Syntax: `mkdir [path]` (erzeugt Ordner mit Pfad `[path]`)
- Beispiel:
  - `mkdir neuer-ordner` (relativ zum aktuellen Pfad)
  - `mkdir -p /tmp/neuer-ordner/tmp1/tmp2/`
- `-p` erstellt alle Ordner auf dem Pfad, die noch nicht existieren

make directory
Verzeichnisse erschaffen

`mkdir` erstellt Ordner auf dem Rechner

- Syntax: `mkdir [path]` (erzeugt Ordner mit Pfad `[path]`)
- Beispiel:
  - `mkdir neuer-ordner` (relativ zum aktuellen Pfad)
  - `mkdir -p /tmp/neuer-ordner/tmp1/tmp2/`
- `-p` erstellt alle Ordner auf dem Pfad, die noch nicht existieren

remove directory

Löschen von leeren Ordnern.

- Befehl ist weniger mächtig als man erwartet
- Vorsicht ist geboten, auch leere Verzeichnisse können wichtig sein!!
- Leere Ordner in dem leeren Ordner werden auch gelöscht
- für nicht-leere Ordner siehe `rm`

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Input, Output und Error

- Jedes Programm hat drei Datenströme:
  1) `stdin` (Standard Input [number: 0])
  2) `stdout` (Standard Output [number: 1])
  3) `stderr` (Standard Error Output [number: 2])
- Bsp.: `echo`
  - `echo` gibt einen eingegebenen Text zurück
  - Syntax: `echo [text]`
  - dabei ist `[text]` `Eingabe` und die Konsole `stdout`
  - Bsp.`stdin`: cat
  - Bsp. `stdout`: echo Hello World | cat
  - Bsp. `stderr`: cat foo.bar (Datei existiert nicht)

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl → Eingabe anderer Befehl
  - Bsp.: `cal | less`

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl → Eingabe anderer Befehl
  - Bsp.: `cal | less`

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl → Eingabe anderer Befehl
  - Bsp.: `cal | less`

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl → Eingabe anderer Befehl
  - Bsp.: `cal | less`

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl → Eingabe anderer Befehl
  - Bsp.: `cal | less`

Pipe, < und >

Umleiten der Ein- und Ausgabe

- < und > laden und schreiben Dateien
  - > Schreiben in eine Datei
  - Bsp.: `cal > test.txt`
  - < Laden aus einer Datei
  - Bsp.: `less < test.txt`
- Pipe |
  - Ausgabe von Befehl $\rightarrow$ Eingabe anderer Befehl
  - Bsp.: `cal | less`

concatenate
Was eine Katze mit Linux zu tun hat

`cat` konkateniert und zeigt die Inhalte von Dateien

- An sich nicht mächtig, sehr einfacher Befehl
- Parameter (Default: -u):
  - `-n` Nummeriert die ausgegebenen Zeilen
  - `-e` Fügt am Ende jeder Zeile ein $ hinzu
- Bsp.: `cat -n output.txt output2.txt`

concatenate
Was eine Katze mit Linux zu tun hat

`cat` konkateniert und zeigt die Inhalte von Dateien

- An sich nicht mächtig, sehr einfacher Befehl
- Parameter (Default: -u):
  - `-n` Nummeriert die ausgegebenen Zeilen
  - `-e` Fügt am Ende jeder Zeile ein $ hinzu
- Bsp.: `cat -n output.txt output2.txt`

less
Even more than more

`less` ist intelligenter Nachfolger von `more`

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - `:n` ruft das nächste Dokument auf (wenn less mehrere bekommen hat)
  - `:p` ruft das vorherige Dokumente auf
  - `:f` die Anzahl der Zeilen und die Dateigröße werden angezeigt

less
Even more than `more`

`less` ist intelligenter Nachfolger von `more`

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - `:n` ruft das nächste Dokument auf (wenn less mehrere bekommen hat)
  - `:p` ruft das vorherige Dokumente auf
  - `:f` die Anzahl der Zeilen und die Dateigröße werden angezeigt

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)
  - :p ruft das vorherige Dokumente auf
  - :f die Anzahl der Zeilen und die Dateigröße werden angezeigt

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)
  - :p ruft das vorherige Dokumente auf
  - :f die Anzahl der Zeilen und die Dateigröße werden angezeigt

globally search a regular expression and print
Gonna catch em' Strings

`grep` ermöglicht die Suche in einer Eingabe

- `-i` ignoriert Groß- und Kleinschreibung bei der Suche

Beispiel:

- `ls | grep 'Documents'`
- `cat example.txt | grep -i 'TeSt'`

nano

Ermöglicht das bearbeiten von Textdateien im Terminal.

- minimalistisch
- leichte Bedienung
- bietet alle Standardfunktionen eines Texteditors

Umfangreichere Alternative stellt `vim` dar.

copy & move
Die Kunst des Klonens und Umbenennens

`cp` und `mv` sind neben `ls` und `cd` wohl die wichtigsten Befehle in Linux.

- Dateiname muss bei beiden Befehlen nicht gleich bleiben
- `mv [path1] [path2]`
  - verschiebt Datei von `[path1]` zu `[path2]`
  - Parameter (Default: -f):
    - `-i` (interactive) fragt vor Überschreiben
    - `-f` (force) überschreibt **immer**
    - `-n` (no-clobber) überschreibt **niemals**
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern

copy & move
Die Kunst des Klonens und Umbenennens

`cp` und `mv` sind neben `ls` und `cd` wohl die wichtigsten Befehle in Linux.

- Dateiname muss bei beiden Befehlen nicht gleich bleiben
- `mv [path1] [path2]`
  - verschiebt Datei von `[path1]` zu `[path2]`
  - Parameter (Default: -f):
    - `-i` (interactive) fragt vor Überschreiben
    - `-f` (force) überschreibt **immer**
    - `-n` (no-clobber) überschreibt **niemals**
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern

copy & move
Die Kunst des Klonens und Umbenennens

`cp` und `mv` sind neben `ls` und `cd` wohl die wichtigsten Befehle in Linux.

- Dateiname muss bei beiden Befehlen nicht gleich bleiben
- `mv [path1] [path2]`
  - verschiebt Datei von `[path1]` zu `[path2]`
  - Parameter (Default: -f):
    - `-i` (interactive) fragt vor Überschreiben
    - `-f` (force) überschreibt **immer**
    - `-n` (no-clobber) überschreibt **niemals**
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern

copy & move
Die Kunst des Klonens und Umbenennens

`cp` und `mv` sind neben `ls` und `cd` wohl die wichtigsten Befehle in Linux.

- Dateiname muss bei beiden Befehlen nicht gleich bleiben
- `mv [path1] [path2]`
  - verschiebt Datei von `[path1]` zu `[path2]`
  - Parameter (Default: -f):
    - `-i` (interactive) fragt vor Überschreiben
    - `-f` (force) überschreibt **immer**
    - `-n` (no-clobber) überschreibt **niemals**
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern

remove
Flutsch! Und weg!

Immer wieder im Leben kommt eine Zeit in der man etwas Ballast abwerfen möchte. `rm` hilft dabei

- Syntax: `rm [option] [file]`
- Parameter (Default: -f):
  - `-f` (force) es wird nicht nachgefragt und einfach alles gelöscht
  - `-i` Nachfragen bei jeder Löschung
  - `-r` Löschung rekursiv → (Ordner und Unterordner)
  - `-d` leere Verzeichnisse löschen

remove
Flutsch! Und weg!

Immer wieder im Leben kommt eine Zeit in der man etwas Ballast abwerfen möchte. `rm` hilft dabei

- Syntax: `rm [option] [file]`
- Parameter (Default: -f):
  - `-f` (force) es wird nicht nachgefragt und einfach alles gelöscht
  - `-i` Nachfragen bei jeder Löschung
  - `-r` Löschung rekursiv → (Ordner und Unterordner)
  - `-d` leere Verzeichnisse löschen

remove
Flutsch! Und weg!

Immer wieder im Leben kommt eine Zeit in der man etwas Ballast abwerfen möchte. `rm` hilft dabei

- Syntax: `rm [option] [file]`
- Parameter (Default: -f):
  - `-f` (force) es wird nicht nachgefragt und einfach alles gelöscht
  - `-i` Nachfragen bei jeder Löschung
  - `-r` Löschung rekursiv $\rightarrow$ (Ordner und Unterordner)
  - `-d` leere Verzeichnisse löschen

remove
Flutsch! Und weg!

Immer wieder im Leben kommt eine Zeit in der man etwas Ballast abwerfen möchte. `rm` hilft dabei

- Syntax: `rm [option] [file]`
- Parameter (Default: -f):
  - `-f` (force) es wird nicht nachgefragt und einfach alles gelöscht
  - `-i` Nachfragen bei jeder Löschung
  - `-r` Löschung rekursiv $\rightarrow$ (Ordner und Unterordner)
  - `-d` leere Verzeichnisse löschen

tape archiver
Archivieren

Zum entpacken von `.tar`, `.tar.gz`, `.zip`, etc. via Konsole.

- Syntax entpacken: `tar [option] [path]`
- Syntax verpacken: `tar [option] [path-archiv] [path-files]`
- Parameter:
  - `-c` (create) erzeugt neues Archiv
  - `-x` (extract) extrahieren einer Datei
  - `-v` (verbose) Fortschritt auflisten
  - `-z` (gzip format) komprimieren als `.gz`, etc.
  - `-f` erzeugt beim Entpacken einen Ordner mit Namen von Archiv
- mit cat können sogar 2 Archive zusammengeführt werden
- Bsp.: `tar -xvf test.tar`

tape archiver
Archivieren

Zum entpacken von `.tar`, `.tar.gz`, `.zip`, etc. via Konsole.

- Syntax entpacken: `tar [option] [path]`
- Syntax verpacken: `tar [option] [path-archiv] [path-files]`
- Parameter:
  - `-c` (create) erzeugt neues Archiv
  - `-x` (extract) extrahieren einer Datei
  - `-v` (verbose) Fortschritt auflisten
  - `-z` (gzip format) komprimieren als `.gz`, etc.
  - `-f` erzeugt beim Entpacken einen Ordner mit Namen von Archiv
- mit cat können sogar 2 Archive zusammengeführt werden
- Bsp.: `tar -xvf test.tar`

tape archiver
Archivieren

Zum entpacken von `.tar`, `.tar.gz`, `.zip`, etc. via Konsole.

- Syntax entpacken: `tar [option] [path]`
- Syntax verpacken: `tar [option] [path-archiv] [path-files]`
- Parameter:
  - `-c` (create) erzeugt neues Archiv
  - `-x` (extract) extrahieren einer Datei
  - `-v` (verbose) Fortschritt auflisten
  - `-z` (gzip format) komprimieren als `.gz`, etc.
  - `-f` erzeugt beim Entpacken einen Ordner mit Namen von Archiv
- mit cat können sogar 2 Archive zusammengeführt werden
- Bsp.: `tar -xvf test.tar`

tape archiver
Archivieren

Zum entpacken von `.tar`, `.tar.gz`, `.zip`, etc. via Konsole.

- Syntax entpacken: `tar [option] [path]`
- Syntax verpacken: `tar [option] [path-archiv] [path-files]`
- Parameter:
  - `-c` (create) erzeugt neues Archiv
  - `-x` (extract) extrahieren einer Datei
  - `-v` (verbose) Fortschritt auflisten
  - `-z` (gzip format) komprimieren als `.gz`, etc.
  - `-f` erzeugt beim Entpacken einen Ordner mit Namen von Archiv
- mit cat können sogar 2 Archive zusammengeführt werden
- Bsp.: `tar -xvf test.tar`

tape archiver
Archivieren

Zum entpacken von `.tar`, `.tar.gz`, `.zip`, etc. via Konsole.

- Syntax entpacken: `tar [option] [path]`
- Syntax verpacken: `tar [option] [path-archiv] [path-files]`
- Parameter:
  - `-c` (create) erzeugt neues Archiv
  - `-x` (extract) extrahieren einer Datei
  - `-v` (verbose) Fortschritt auflisten
  - `-z` (gzip format) komprimieren als `.gz`, etc.
  - `-f` erzeugt beim Entpacken einen Ordner mit Namen von Archiv
- mit cat können sogar 2 Archive zusammengeführt werden
- Bsp.: `tar -xvf test.tar`

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - Übersicht über alle Prozesse und verbrauchte Ressourcen
  - deutlich leichtere Bedienung und bessere Übersicht
  - bietet mehr Interaktionen

hisham **t**able **o**f **p**rocesses
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - Übersicht über alle Prozesse und verbrauchte Ressourcen
  - deutlich leichtere Bedienung und bessere Übersicht
  - bietet mehr Interaktionen

hisham **t**able **o**f **p**rocesses
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - Übersicht über alle Prozesse und verbrauchte Ressourcen
  - deutlich leichtere Bedienung und bessere Übersicht
  - bietet mehr Interaktionen

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - Übersicht über alle Prozesse und verbrauchte Ressourcen
  - deutlich leichtere Bedienung und bessere Übersicht
  - bietet mehr Interaktionen

&
Backstage

Bei Angabe dieses Operators am Ende eines Prozess-Aufrufs, wird der Prozess in den Hintergrund verschoben

- Prozess blockiert Shell nicht $\rightarrow$ weiteres Arbeiten möglich
- Ausgabe wird aber weiterhin auf Shell geschrieben

Achtung!! & $\equiv$ Prozess in Hintergrund; && $\equiv$ Prozesse hintereinander ausführen (p1 && p2 : p2 wird ausgeführt, wenn p1 erfolgreich)

ip addr

Meist wurde `ifconfig` benutzt um sich seine lokalen IP-Adressen des Rechners anzeigen zu lassen. Dieser Befehl ist allerdings ein wenig obsolet und wurde ersetzt durch `ip addr`.

- Um lokale IP-Adresse zu erfahren, ohne über Router zu gehen
- leider etwas unübersichtlich
  - Gegenmaßnahme: `ip addr | grep inet`

super user do
Mit großer Macht kommt große Verantwortung

`sudo` führt einen Befehl mit administrativer Berechtigung aus `sudo [command]`. **Achtung!!** `sudo` funktioniert nur, wenn Benutzer in `root`-Gruppe ist.

- Führt man Befehle als Root aus, sollte man vorsichtig sein
- Parameter:
  - `-b` (background) führt Befehl im Hintergrund aus [kommt später]
  - `-e` (edit) öffnet Datei zum editieren, erstellt temp. Backup
  - `-s` (shell) öffnet eine Shell
  - `-u [user]` führt Befehl als `[user]` aus

super user do
Mit großer Macht kommt große Verantwortung

`sudo` führt einen Befehl mit administrativer Berechtigung aus `sudo [command]`. **Achtung!!** `sudo` funktioniert nur, wenn Benutzer in `root`-Gruppe ist.

- Führt man Befehle als Root aus, sollte man vorsichtig sein
- Parameter:
  - `-b` (background) führt Befehl im Hintergrund aus [kommt später]
  - `-e` (edit) öffnet Datei zum editieren, erstellt temp. Backup
  - `-s` (shell) öffnet eine Shell
  - `-u [user]` führt Befehl als `[user]` aus

super user do
Mit großer Macht kommt große Verantwortung

`sudo` führt einen Befehl mit administrativer Berechtigung aus `sudo [command]`. **Achtung!!** `sudo` funktioniert nur, wenn Benutzer in `root`-Gruppe ist.

- Führt man Befehle als Root aus, sollte man vorsichtig sein
- Parameter:
  - `-b` (background) führt Befehl im Hintergrund aus [kommt später]
  - `-e` (edit) öffnet Datei zum editieren, erstellt temp. Backup
  - `-s` (shell) öffnet eine Shell
  - `-u [user]` führt Befehl als `[user]` aus

super user do
Mit großer Macht kommt große Verantwortung

`sudo` führt einen Befehl mit administrativer Berechtigung aus `sudo [command]`. **Achtung!!** `sudo` funktioniert nur, wenn Benutzer in `root`-Gruppe ist.

- Führt man Befehle als Root aus, sollte man vorsichtig sein
- Parameter:
  - `-b` (background) führt Befehl im Hintergrund aus [kommt später]
  - `-e` (edit) öffnet Datei zum editieren, erstellt temp. Backup
  - `-s` (shell) öffnet eine Shell
  - `-u [user]` führt Befehl als `[user]` aus

apt

Dieser Paketmanager wird nur in Debian basierten Systemen benutzt. Es gibt aber auch noch weitere, z.B. Pacman für Arch basierte Systeme.

- Wichtigsten Befehle:
  1. `sudo apt install [pkg...]`
  2. `sudo apt update`
  3. `sudo apt [upgrade | dist-upgrade]`
  4. `sudo apt remove [pkg...]`
  5. `sudo apt search [pkg...]`

man page
Wenn man mal nicht weiter weiß

Die Man-Page ist die Benutzeranleitung des Systems, mit Anleitungen zur Benutzung der installierten Programme

- Sektionen, die besonders interessant sind:
  - `NAME`, `SYNOPSIS`, `DESCRIPTION`, `OPTIONS`, `EXAMPLE`

poweroff und reboot
Wenn man mal das real life genießen will

- `poweroff` (Herunterfahren des Systems)
- `reboot` (Neustart des Systems)