Linux Basics

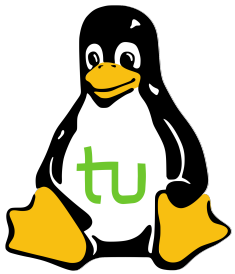
— Das Terminal —

Jan-Marius Lenk, Christoph Parnitzke, Yannick Bungers

Free and Open Source Software AG
Fakultät für Informatik
5. Dezember 2016

Inhaltsverzeichnis

Linux Philosophie



- Philosophie besteht aus drei Punkten:
  - 1. Schreibe Programme, die nur eine Sache tun und dies erfolgreich (DOTADIW)
  - 2. Schreibe Programme, die Kolaboration ermöglichen
  - 3. Schreibe Programme, die mit Text arbeiten, denn dies ist universell
- Ist auch heute noch Kernaussage

Terminal vs. Shell

- Terminal ist zeilenweise Eingabe von Befehlen
  1) TTY (teletype) sind Terminals, erreichbar via STRG+ALT+F[1-7]
  2) VTerm: virtueller Terminal in grafischer Oberfläche
  3) Shell: interpretiert Eingabe des Benutzers

switch user

- Wechsel des Benutzers innerhalb der Konsole
- Ohne Parameter wird versucht sich als `root` einzuloggen
- Ansonsten `su [user]` einloggen als `[user]`

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
- Wird im Vortrag `[nur Root]` angegeben, muss der Befehl mit `sudo` gestartet werden

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
- Wird im Vortrag `[nur Root]` angegeben, muss der Befehl mit `sudo` gestartet werden

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
- Wird im Vortrag `[nur Root]` angegeben, muss der Befehl mit `sudo` gestartet werden

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
- Wird im Vortrag `[nur Root]` angegeben, muss der Befehl mit `sudo` gestartet werden

apt-get vs. apt vs. aptitude

Diese Paket Manager werden nur in Debian basierten Systemen benutzt. Es gibt aber auch noch weitere, z.B. Pacman für Arch basierte Systeme. `[nur Root]`

- Paket Management: `apt` Kurzform für `apt-get` und `apt-cache`
- Wichtigsten Befehle:
  1. `apt install [pkg...]`
  2. `apt update`
  3. `apt [upgrade | dist-upgrade | full-upgrade]`
  4. `apt remove [pkg...]`
- `aptitude` ist etwas grafischer und hat Zusatzfunktionen

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 und BUGS)`
- **Wie werden Einträge gesucht?**
  - ausführbare Programme und Shell Kommandos
  - Systemaufrufe (vom Kernel bereit gestellt)
  - Bibliotheksaufrufe (Programm-Bibliotheken)
  - spezielle Dateien (in `/dev`)
  - Dateiformate und Konventionen (`/etc/passwd`)
  - Spiele
  - Sonstiges (Macropakete und Konventionen)
  - Systemadministrations Kommandos `[nur Root]`
  - Kernel Routinen (kein Standard)

Aufgabe 1

Nun sollt ihr euch ein wenig mit der `man`-Page und dem Paket-Management vertraut machen. Zur Bearbeitung dieser Aufgabe habt ihr 10 Minuten Zeit.

1) führt ein Update der Quellen durch; `apt update`
2) installiert das Paket `funny-manpages`
   - benutzt dazu `apt install`
   - `sudo` nicht vergessen!!
3) betrachtet folgende `man`-Pages:
   - `su`, `apt-get`, `party`, `t`, `tm`, `flog`
4) (optional) `funny-manpages` entfernen, wer mag
   - benutzt dazu `apt remove`

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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) 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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) 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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) 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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) 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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) 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` (reversiv) umgekehrte Reihenfolge
  - `-m` Namen durch Kommata getrennt
  - `-R` (rekursiv) inkl. Unterverzeichnisse
- Bsp.: `ls -ahl` und `ls -ah`

change directory
Die kleine Form der Teleportation

`cd` bietet 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` bietet 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` bietet 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` bietet 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` bietet 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` bietet 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` bietet 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

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 Parent-Directory, falls es nicht existiert
- `--mode=[mode]` Modus mit dem der Ordner erstellt wird (siehe `chmod`)

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 Parent-Directory, falls es nicht existiert
- `--mode=[mode]` Modus mit dem der Ordner erstellt wird (siehe `chmod`)

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

Gruppenverwaltung

- `groups [username]` (zeigt Gruppen von Benutzer an)
- `groupadd [groupname]` (legt Gruppe an) `[nur Root]`
- `groupdel [groupname]` (löscht Gruppe) `[nur Root]`
- `groupmod [option] [groupname]` (Bsp.: `--new-name [Name]`) `[nur Root]`
- `usermod -aG [username] [groupname]` (Benutzer zu Gruppe hinzufügen) [nur Root]
- `deluser [username] [groupname]` (Benutzer aus Gruppe entfernen) `[nur Root]`

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch a als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch `a` als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch `a` als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch `a` als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch a als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change mode
Mein Ordner gehört mir

`chmod` ändert die Zugriffsrechte von Dateien und Ordnern

- Syntax: `chmod [mode] [path]`
- Beispiel (`/home/[username]/neuer-ordner/`):
  - aktuelle Rechte: `u=rwx g=r-x o=r-x`
  - `chmod g=rwx ~/neuer-ordner` (wir geben der Gruppe Schreib-Recht)
  - `chmod o-r ~/neuer-ordner` (wir nehmen allen anderen das Lese-Recht)
  - neue Rechte: `u=rwx g=rwx o=--x`
  - es gibt auch a als Argument, was für all steht
  - Bsp. Schreib-Recht von `group` und `others` nehmen: chmod g-x,o-x [path]

change owner

Ändern des Besitzers von Ordern und Dateien.

- Zu jeder Datei/Ordner gehören 2 Besitzer:
  - ein Nutzer
  - eine Gruppe
- `chown [user] [file] [file]` gehört jetzt `[user]`
- `chown [user]:[group] [file] [file]` gehört jetzt zu `[group]` und `[user]`
- Parameter `-R` ändert die Zugehörigkeit rekursiv

change owner

Ändern des Besitzers von Ordern und Dateien.

- Zu jeder Datei/Ordner gehören 2 Besitzer:
  - ein Nutzer
  - eine Gruppe
- `chown [user] [file] [file]` gehört jetzt `[user]`
- `chown [user]:[group] [file] [file]` gehört jetzt zu `[group]` und `[user]`
- Parameter `-R` ändert die Zugehörigkeit rekursiv

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

Aufgabe 2)

In der zweiten Aufgabe sollt ihr euch mit der Verwaltung von Ordnern beschäftigen. Zur Bearbeitung stehen euch 15 Minuten zur verfügung.

- wechselt in das Verzeichnis `/tmp`
- erstellt den Ordner `test`
- guckt euch die Rechte des Ordners an
- entfernt die Rechte für `group` und `other` des Ordners
- setzt als neuen Owner des Ordners `root [nur Root]`
- versucht den Ordner zu entfernen (ohne `sudo`). Was passiert?
- wechselt nun wieder den Owner zurück `[nur Root]`
- entfernt den Ordner

Tipp: `cd`, `ls`, `mkdir`, `chmod`, `chown`, `rmdir`

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` und `stderr`
  - Bsp. stdin: cat

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` und `stderr`
  - Bsp. stdin: cat

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` und `stderr`
  - Bsp. stdin: cat

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` und `stderr`
  - Bsp. stdin: cat

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

|, < 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.: `date | less`
- fancy stuff:
  - Umleiten von stderr: 2> statt >
  - also durch Angabe von Stream Nummer
  - Bsp.: `ls -r > output.txt 2>&1`

more
Less than everything

`more` ist ein Filereader

- Angucken von Datei ohne Editieren
- Arbeitet auf Standardausgabe
- Beherrscht alle dem System bekannten Codierungen
- Navigierung ist allerdings ein wenig hinderlich
- Grundlage zu less

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - /[pattern] Durchsucht das Dokument nach angegebenen Muster
  - ?[pattern]' ist wie '/[pattern] nur rückwärts
  - &[pattern] zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt n das nächste Vorkommen
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

less
Even more than `more`

`less` ist intelligenter Nachfolger von `more`

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - `/[pattern]` Durchsucht das Dokument nach angegebenen Muster
  - `?[pattern]' ist wie '/[pattern]` nur rückwärts
  - `&[pattern]` zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt `n` das nächste Vorkommen
  - `:n` ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - /[pattern] Durchsucht das Dokument nach angegebenen Muster
  - ?[pattern]' ist wie '/[pattern] nur rückwärts
  - &[pattern] zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt n das nächste Vorkommen
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - /[pattern] Durchsucht das Dokument nach angegebenen Muster
  - ?[pattern]' ist wie '/[pattern] nur rückwärts
  - &[pattern] zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt n das nächste Vorkommen
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - /[pattern] Durchsucht das Dokument nach angegebenen Muster
  - ?[pattern]' ist wie '/[pattern] nur rückwärts
  - &[pattern] zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt n das nächste Vorkommen
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

less
Even more than more

less ist intelligenter Nachfolger von more

- unterstützt Scrolling
- Navigation:
  - Pfeil- und Bildlauftasten
  - /[pattern] Durchsucht das Dokument nach angegebenen Muster
  - ?[pattern]' ist wie '/[pattern] nur rückwärts
  - &[pattern] zeigt nur Zeilen, mit Muster
  - wird pattern benutzt, dann zeigt n das nächste Vorkommen
  - :n ruft das nächste Dokument auf (wenn less mehrere bekommen hat)

Aufgabe 3)

In dieser Aufgabe sollt ihr euch mit der Umleitung der Standardausgabe beschäftigen. Für die Bearbeitung habt ihr 5 Minuten.

- wechselt in euer Home-Verzeichnis
- lasst euch die Inhalte reversiv ausgeben
- benutzt die Ausgabe als Eingabe für `cowsay` (Pipe-Operator)
- lasst euch nun diese Ausgabe in die Datei `KuhSagtHome.txt` schreiben
- seht euch den Inhalt der Datei an

Tipp: `cd`, `ls`, `cowsay`, `|`, `>`, `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
  - `-s` Unterdrückt Ausgabe von sich wiederholenden Leerzeichen
- Bsp.: `cat -n output.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
  - `-s` Unterdrückt Ausgabe von sich wiederholenden Leerzeichen
- Bsp.: `cat -n output.txt`

head & tail
Dateien von Kopf bis Fuß

Ähnlich zu `cat`, sind `head` und `tail` einfach Textdatei Betrachter. Allerdings beschränken sich diese auf bestimmte Teile der Datei.

- `head` gibt ersten 10 Zeilen der angegebenen Datei aus
- `tail` gibt letzten 10 Zeilen der angegebenen Datei aus
- Parameter (Default: -q):
  - `head`:
    - `-n [val]` gibt ersten `[val]` Zeilen aus
  - `tail`:
    - `-n [val]` analog zu `head`
    - `-f` fügt weitere Zeilen zur Ausgabe hinzu, falls Inhalt weiter anwächst

head & tail
Dateien von Kopf bis Fuß

Ähnlich zu `cat`, sind `head` und `tail` einfach Textdatei Betrachter. Allerdings beschränken sich diese auf bestimmte Teile der Datei.

- `head` gibt ersten 10 Zeilen der angegebenen Datei aus
- `tail` gibt letzten 10 Zeilen der angegebenen Datei aus
- Parameter (Default: -q):
  - `head`:
    - `-n [val]` gibt ersten `[val]` Zeilen aus
  - `tail`:
    - `-n [val]` analog zu `head`
    - `-f` fügt weitere Zeilen zur Ausgabe hinzu, falls Inhalt weiter anwächst

head & tail
Dateien von Kopf bis Fuß

Ähnlich zu `cat`, sind `head` und `tail` einfach Textdatei Betrachter. Allerdings beschränken sich diese auf bestimmte Teile der Datei.

- `head` gibt ersten 10 Zeilen der angegebenen Datei aus
- `tail` gibt letzten 10 Zeilen der angegebenen Datei aus
- Parameter (Default: -q):
  - `head`:
    - `-n [val]` gibt ersten `[val]` Zeilen aus
  - `tail`:
    - `-n [val]` analog zu `head`
    - `-f` fügt weitere Zeilen zur Ausgabe hinzu, falls Inhalt weiter anwächst

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

Sortieren und Suchen

Auch wenn man beim Sortieren erst einmal an das Sortieren von Dateien denkt, bezieht sich das Sortieren auf den Inhalt von Dateien. Das Suchen bezieht sich auf das Finden von Dateien.

- `sort [option] [File]` (liest `File` und sortiert Inhalt)
  - `-g` ist Default. generic sort
  - `-r` Ausgabe umdrehen
  - `-c` Check ob Inhalt von Datei sortiert ist
  - Bsp.: `sort -r output.txt`
- `find [path] [pattern]` (suchen nach Datei via Muster) `-P` Default
  - **Achtung!!** Suche kann lange dauern
  - Bsp.: `find -name output.txt`

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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**
    - `--backup` erstellt Backup vor Überschreiben
- `cp [path1] [path2]`
  - kopiert Datei von `[path1]` zu `[path2]`
  - Parameter (dieselben wie `mv`):
    - `-r` (recursive) ermöglicht verschieben von Ordnern
    - `-s` (symbolic-link) erstellt symbolische Links, statt Dateien

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
- `rm` löscht nicht entgültig, Daten sind nicht unleserlich
- soll entgültig gelöscht werden, dann benutze `shred`

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
- `rm` löscht nicht entgültig, Daten sind nicht unleserlich
- soll entgültig gelöscht werden, dann benutze `shred`

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
- `rm` löscht nicht entgültig, Daten sind nicht unleserlich
- soll entgültig gelöscht werden, dann benutze `shred`

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
- `rm` löscht nicht entgültig, Daten sind nicht unleserlich
- soll entgültig gelöscht werden, dann benutze `shred`

Aufgabe 4)

Diese Aufgabe soll euch ein Gefühl für den Umgang mit Dateien geben. Für die Bearbeitung habt ihr 15 Minuten.

- lasst euch alle Dateien in `/usr/share` ausgeben, die dem folgenden RegEx entsprechen `"*.txt"` und schreibt die Ausgabe in `/home/[username]/txt-files.txt`
- schaut euch nun den Inhalt an
- sucht nach eurer Datei `KuhSagtHome.txt` und schreibt das Ergebnis in `kuh-sagt-path.txt` (Suche in `/home`)
- verbindet nun den Inhalt von `txt-files.txt` und `kuh-sagt-path.txt` und schreibt die Ausgabe in `cat-files-kuh.txt`
- betrachtet das Ende der Datei `cat-files-kuh.txt`. Was fällt auf?
- sortiert den Inhalt von `cat-files-kuh.txt` numerisch und schreibt die Ausgabe in `sort-cat-kuh.txt`
- betrachtet den Anfang von `sort-cat-kuh.txt`. Was hat sich verändert?
- löscht nun alle eben erstellten Dateien

Tipp: `find`, `less` bzw. `more`, `cat`, `tail`, `head`, `sort`, `rm` bzw. `shred`

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)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

hisham table of processes
interaktiver Prozessmanager

- `table of processes` (bereits installtiert)
  - `-h` um Befehle anzeigen zu lassen
- `htop` (muss installiert werden)
  - deutlich leichtere Bedienung
- Ansicht:
  - Auslastung der einzelnen CPU-Kerne und des Arbeitsspeichers
  - Anteil vom Swap und bisherige Laufzeit des Systems
  - PID (Prozess ID), User, PRI (Priorität), NI (Nice-Wert)
  - VIRT und RES (benutzer virtueller und physischer Speicher)
  - S (State; S $\in$ {S[leeping], R[unning], D[isk sleep], Z[ombie], T[raced/suspended], W[Paging]})
  - CPU% und MEM% (genutzter CPU und Arbeitsspeicher Anteil) und TIME (bisherige Laufzeit)

nice
nette Programme

Ausführen von Programmen mit modifizierter sheduling Priorität p, p $\in$ [-20,19]. -20 ist die höchste Priorität.

- Default: p = 0; negative Werte nur als Root
- Syntax: `nice [option] [command] [arg] ...`
- Parameter (Default: -n 10):
  - `-n (--adjustment=N)` aufaddieren von Integer N auf aktuellen Nice-Wert (Default: N=10)
- Bsp. `ping`:
  - Syntax: `ping [option] [addr]`
  - Parameter: `-c [num]` (setze Anzahl der Pings), `-W [time]` (setze Timeout)
  - `nice -n 10 ping foss-ag.de`

nice
nette Programme

Ausführen von Programmen mit modifizierter sheduling Priorität p, p $\in$ [-20,19]. -20 ist die höchste Priorität.

- Default: p = 0; negative Werte nur als Root
- Syntax: `nice [option] [command] [arg] ...`
- Parameter (Default: -n 10):
  - `-n (--adjustment=N)` aufaddieren von Integer N auf aktuellen Nice-Wert (Default: N=10)
- Bsp. `ping`:
  - Syntax: `ping [option] [addr]`
  - Parameter: `-c [num]` (setze Anzahl der Pings), `-W [time]` (setze Timeout)
  - `nice -n 10 ping foss-ag.de`

nice
nette Programme

Ausführen von Programmen mit modifizierter sheduling Priorität p, p $\in$ [-20,19]. -20 ist die höchste Priorität.

- Default: p = 0; negative Werte nur als Root
- Syntax: `nice [option] [command] [arg] ...`
- Parameter (Default: -n 10):
  - `-n (--adjustment=N)` aufaddieren von Integer N auf aktuellen Nice-Wert (Default: N=10)
- Bsp. `ping`:
  - Syntax: `ping [option] [addr]`
  - Parameter: `-c [num]` (setze Anzahl der Pings), `-W [time]` (setze Timeout)
  - `nice -n 10 ping foss-ag.de`

renice und nohup

`renice` ermöglicht nachträgliches Einstellen des Nice-Wertes eines bereits laufenden Prozesses. `nohup` ermöglicht es ein Programm weiterlaufen zu lassen, auch wenn der Benutzer sich ausloggt.

- `renice`
  - Syntax: `renice [-n] priority [-g | -p | -u] identifier`
  - `-n [num]` ist neuer Nice-Wert
  - `-g [group-id]`, `-p [process-id]`, `-u [user-id]`
- `nohup`
  - Syntax: `nohup [commando] [args]`
  - Ausgabe wird dabei in eine `nohup.txt` Datei im Home-Directory geschrieben (umlenken möglich)

renice und nohup

`renice` ermöglicht nachträgliches Einstellen des Nice-Wertes eines bereits laufenden Prozesses. `nohup` ermöglicht es ein Programm weiterlaufen zu lassen, auch wenn der Benutzer sich ausloggt.

- `renice`
  - Syntax: `renice [-n] priority [-g | -p | -u] identifier`
  - `-n [num]` ist neuer Nice-Wert
  - `-g [group-id]`, `-p [process-id]`, `-u [user-id]`
- `nohup`
  - Syntax: `nohup [commando] [args]`
  - Ausgabe wird dabei in eine `nohup.txt` Datei im Home-Directory geschrieben (umlenken möglich)

renice und nohup

`renice` ermöglicht nachträgliches Einstellen des Nice-Wertes eines bereits laufenden Prozesses. `nohup` ermöglicht es ein Programm weiterlaufen zu lassen, auch wenn der Benutzer sich ausloggt.

- `renice`
  - Syntax: `renice [-n] priority [-g | -p | -u] identifier`
  - `-n [num]` ist neuer Nice-Wert
  - `-g [group-id]`, `-p [process-id]`, `-u [user-id]`
- `nohup`
  - Syntax: `nohup [commando] [args]`
  - Ausgabe wird dabei in eine `nohup.txt` Datei im Home-Directory geschrieben (umlenken möglich)

&
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)

Texteditoren - 1: ed, nano, pico
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist ed (release 1970; Grundlage für grep).

- pico (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- nano (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von pico
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- nano und pico sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 1: ed, nano, pico
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist ed (release 1970; Grundlage für grep).

- pico (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- nano (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von pico
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- nano und pico sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 1: ed, nano, pico
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist ed (release 1970; Grundlage für grep).

- pico (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- nano (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von pico
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- nano und pico sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 1: `ed`, `nano`, `pico`
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist `ed` (release 1970; Grundlage für `grep`).

- `pico` (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- `nano` (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von `pico`
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- `nano` und `pico` sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 1: `ed`, `nano`, `pico`
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist `ed` (release 1970; Grundlage für `grep`).

- `pico` (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- `nano` (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von `pico`
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- `nano` und `pico` sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 1: `ed`, `nano`, `pico`
You would (not) love Ed!!

Das Bearbeiten von Textdateien ist in der Konsole schwierig.
Ein Beispiel ist `ed` (release 1970; Grundlage für `grep`).

- `pico` (release 1992):
  - sehr minimalistischer Editor
  - reine Textverarbeitung, mehrere Dokumente möglich, Syntaxhighlighting
  - kein Highlighting Wörtern, kein Text-Splitscreen, keine RegEx-Suche
- `nano` (release 1999):
  - ebenfalls sehr minimalistisch
  - Nachfolger von `pico`
  - bietet mehr Funktionsumfang, z.B.: Text-Splitscreen, RegEx-Suche
- `nano` und `pico` sind sehr minimalistisch, aber nicht zu unterschätzen

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Texteditoren - 2: `vim`, `jed`
Good thing Ed is not around anymore

Wir haben grad die minimaltistischen Editoren betrachtet, nun die etwas mächtigeren.

- `vim` (release 1991):
  - Großmufti unter den Editoren
  - verbesserte Suchfunktionen, mächtiger als `nano` und `pico`
  - Standard bei `git`
- `jed` (release 1992):
  - **der** Editor für Programmierer
  - liefert Templates für verschiedene Anwendungsfälle
  - weniger mächtig als `vim`, aber dennoch viele Features
- Bei der Wahl des Editors ist es wie immer im Leben, Geschmäcker sind verschieden

Aufgabe 5)

In dieser Aufgabe sollt ihr euch mit der Verwaltung von Prozessen beschäftigen. Für die Bearbeitung stehen euch 10 Minuten zur Verfügung.

- ladet euch die Datei `infinite.tar` von `foss-ag.de` herunter
- verschiebt sie nach `/tmp` und entpackt sie dort
- führt den Befehl `python infinite_run.py` im Hintergrund aus mit Nice-Wert 15
- führt `htop` bzw. `top` aus, sucht euren Prozess und merkt euch die `PID`
- verändert den Nice-Wert auf 10 und schaut in `htop` bzw. `top` nach, was passiert ist
- stoppt nun den Prozess mit dem Befehl `kill`, wobei ihr die `PID` angeben müsst

Tipp: `tar`, `mv`, `nice`, `&`, `renice`, `htop` bzw. `top`

shutdown
Wenn man mal das real life genießen will

ermöglicht Ausschalten oder Neustart des Systems `[nur Root]`

- Syntax: `shutdown [options] [time]`
- Parameter (Default: -P):
  - `-P` (poweroff)
  - `-r` (reboot)
  - `-h` (halt)
- `sudo reboot` anstelle von `sudo shutdown -r`
- Bei `reboot` kein Timer möglich

shutdown
Wenn man mal das real life genießen will

ermöglicht Ausschalten oder Neustart des Systems `[nur Root]`

- Syntax: `shutdown [options] [time]`
- Parameter (Default: -P):
  - `-P` (poweroff)
  - `-r` (reboot)
  - `-h` (halt)
- `sudo reboot` anstelle von `sudo shutdown -r`
- Bei `reboot` kein Timer möglich

shutdown
Wenn man mal das real life genießen will

ermöglicht Ausschalten oder Neustart des Systems `[nur Root]`

- Syntax: `shutdown [options] [time]`
- Parameter (Default: -P):
  - `-P` (poweroff)
  - `-r` (reboot)
  - `-h` (halt)
- `sudo reboot` anstelle von `sudo shutdown -r`
- Bei `reboot` kein Timer möglich

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
- Wird benötigt, wenn man sich via `ssh` einloggen will
- leider etwas unübersichtlich
- Syntax: ip [Option] [Argument]
- Parameter:
  - `-4` Anzeigen der IPv4-Adressen
  - `-6` Anzeigen der IPv6-Adressen

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
- Wird benötigt, wenn man sich via `ssh` einloggen will
- leider etwas unübersichtlich
- Syntax: ip [Option] [Argument]
- Parameter:
  - `-4` Anzeigen der IPv4-Adressen
  - `-6` Anzeigen der IPv6-Adressen

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie top, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie `top`, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie `top`, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

nmon

`nmon` dient zur Überwachung der Auslastung der PC Komponenten

- Shortcuts:
  - `c`, `m`, `d` zeigen Auslastung von CPU-Kerne, Arbeitsspeicher, Festplatte an
  - `k` zeigt Kernel Statistiken an (Queue, Forks, Interrupts)
  - `r` Details zum System und Prozessor
  - `l` CPU Auslastung als Graph
  - `n` Auslastung der Netzwerk Schnittstelle
  - `j` Auslastung des Dateisystems (Platz und Aufbau)
  - `t` wie `top`, Auflistung aller laufenden Prozesse
  - `+` bzw. `-` Refresh erhöhen bzw. verringern
  - `H` für Hilfe (Shortcuts auflisten)

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - c ein konstantes Zeichen c
  - . genau **ein** beliebiges Zeichen
  - .* Folge von beliebigen Zeichen (auch keines)
  - * Zeichen vor * ist beliebig oft (auch gar nicht)
  - ? Zeichen vor ? ist **genau** einmal oder gar nicht
  - + Zeichen vor + ist min. einmal
  - {n,m} Zeichen wird n bis m mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - c ein konstantes Zeichen c
  - . genau **ein** beliebiges Zeichen
  - .* Folge von beliebigen Zeichen (auch keines)
  - * Zeichen vor * ist beliebig oft (auch gar nicht)
  - ? Zeichen vor ? ist **genau** einmal oder gar nicht
  - + Zeichen vor + ist min. einmal
  - {n,m} Zeichen wird n bis m mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - c ein konstantes Zeichen c
  - . genau **ein** beliebiges Zeichen
  - .* Folge von beliebigen Zeichen (auch keines)
  - * Zeichen vor * ist beliebig oft (auch gar nicht)
  - ? Zeichen vor ? ist **genau** einmal oder gar nicht
  - + Zeichen vor + ist min. einmal
  - {n,m} Zeichen wird n bis m mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - c ein konstantes Zeichen c
  - . genau **ein** beliebiges Zeichen
  - .* Folge von beliebigen Zeichen (auch keines)
  - * Zeichen vor * ist beliebig oft (auch gar nicht)
  - ? Zeichen vor ? ist **genau** einmal oder gar nicht
  - + Zeichen vor + ist min. einmal
  - {n,m} Zeichen wird n bis m mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - `c` ein konstantes Zeichen `c`
  - `.` genau **ein** beliebiges Zeichen
  - `.*` Folge von beliebigen Zeichen (auch keines)
  - `*` Zeichen vor `*` ist beliebig oft (auch gar nicht)
  - `?` Zeichen vor `?` ist **genau** einmal oder gar nicht
  - `+` Zeichen vor `+` ist min. einmal
  - `{n,m}` Zeichen wird `n` bis `m` mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - `c` ein konstantes Zeichen `c`
  - `.` genau **ein** beliebiges Zeichen
  - `.*` Folge von beliebigen Zeichen (auch keines)
  - `*` Zeichen vor `*` ist beliebig oft (auch gar nicht)
  - `?` Zeichen vor `?` ist **genau** einmal oder gar nicht
  - `+` Zeichen vor `+` ist min. einmal
  - `{n,m}` Zeichen wird `n` bis `m` mal wiederholt

Regular Expression - 1
The art of black magic

Reguläre Ausdrücke sind Muster, die eine Grundstruktur von gesuchten Begriffen angeben. Um Fehlinterpretationen zu vermeiden, sollte der RegEx immer in Anführungszeichen angegeben werden.

- Syntax - 1:
  - `c` ein konstantes Zeichen `c`
  - `.` genau **ein** beliebiges Zeichen
  - `.*` Folge von beliebigen Zeichen (auch keines)
  - `*` Zeichen vor `*` ist beliebig oft (auch gar nicht)
  - `?` Zeichen vor `?` ist **genau** einmal oder gar nicht
  - `+` Zeichen vor `+` ist min. einmal
  - `{n,m}` Zeichen wird `n` bis `m` mal wiederholt

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 2
The art of black magic

- Syntax - 2:
  - {n} Zeichen genau n mal
  - [...] Zeichen kommt in den Klammern vor
  - [...-...] Zeichen aus Zeichenklasse
  - [^...] und [^...-...] verneinte Zeichenklasse
  - ^ Zeilenanfang
  - $ Zeilenende
  - < Wortanfang
  - > Wortende
  - a1|a2 alternative Ausdrücke; a1 oder a2
  - (...) Gruppe von Ausdrücken
  - \ Sonderbedeutung von nächstem Symbol ignorieren. Bsp.: —

Regular Expression - 3
The art of black magic

- Bsp.: `.*p+ython[1-99]*.*|.*\.pyc?`
- Es werden alle Wörter/Dateien gesucht, die:
  - `python` im Namen haben oder mit .py bzw. .pyc aufhören
  - dürfen mit Zahlen zwischen 1 und 99 enden
  - min. ein p am Anfang haben
- Achtung!! bei find muss RegEx dem Pfad entsprechen

Regular Expression - 3
The art of black magic

- Bsp.: `.*p+ython[1-99]*.*|.*\.pyc?`
- Es werden alle Wörter/Dateien gesucht, die:
  - `python` im Namen haben oder mit .py bzw. .pyc aufhören
  - dürfen mit Zahlen zwischen 1 und 99 enden
  - min. ein p am Anfang haben
- Achtung!! bei find muss RegEx dem Pfad entsprechen

Regular Expression - 3
The art of black magic

- Bsp.: `.*p+ython[1-99]*.*|.*\.pyc?`
- Es werden alle Wörter/Dateien gesucht, die:
  - `python` im Namen haben oder mit .py bzw. .pyc aufhören
  - dürfen mit Zahlen zwischen 1 und 99 enden
  - min. ein p am Anfang haben
- Achtung!! bei find muss RegEx dem Pfad entsprechen

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

Durchsucht eine Eingabe nach einem RegEx.
Der Name kommt von den Befehlen g, re und p aus dem Editor Ed Default: -G

	Selektoren (interpretieren Suchwort als)
`-E`	erweiterter RegEx
`-F`	fixed String
`-G`	basic RegEx

	Ausgabe
`-c`	Anzahl passender Zeilen
`-o`	nur gematchte Teile einer Zeile
`-q`	keine Ausgabe, nur Exitcode (0 wenn erfolgreich)

	Kontrolle
`-f [file]`	liest RegEx aus [file]
`-v`	invertiert Matchmaking
`-w`	nur Zeilen, in denen ganze Worte RegEx entsprechen
`-x`	nur Zeilen, die ganz dem RegEx entsprechen

Alias
How to save time!!!!!

Im System kann für einen Befehl ein Alias definiert werden.
Die Datei `.bashrc` wird beim Einloggen des Users geladen.
Vorsicht beim Bearbeiten!!! Bsp.: ll anstatt von 6ls -l

- editiere die Datei /.bashrc
- füge die Zeile `alias ll='ls -l'` hinzu
- lade Datei neu: `source /.bashrc` bzw. neu einloggen
- Tadaa!!!!

Alias
How to save time!!!!!

Im System kann für einen Befehl ein Alias definiert werden.
Die Datei `.bashrc` wird beim Einloggen des Users geladen.
Vorsicht beim Bearbeiten!!! Bsp.: ll anstatt von 6ls -l

- editiere die Datei /.bashrc
- füge die Zeile `alias ll='ls -l'` hinzu
- lade Datei neu: `source /.bashrc` bzw. neu einloggen
- Tadaa!!!!

Alias
How to save time!!!!!

Im System kann für einen Befehl ein Alias definiert werden.
Die Datei `.bashrc` wird beim Einloggen des Users geladen.
Vorsicht beim Bearbeiten!!! Bsp.: ll anstatt von 6ls -l

- editiere die Datei /.bashrc
- füge die Zeile `alias ll='ls -l'` hinzu
- lade Datei neu: `source /.bashrc` bzw. neu einloggen
- Tadaa!!!!

Alias
How to save time!!!!!

Im System kann für einen Befehl ein Alias definiert werden.
Die Datei `.bashrc` wird beim Einloggen des Users geladen.
Vorsicht beim Bearbeiten!!! Bsp.: ll anstatt von 6ls -l

- editiere die Datei /.bashrc
- füge die Zeile `alias ll='ls -l'` hinzu
- lade Datei neu: `source /.bashrc` bzw. neu einloggen
- Tadaa!!!!

Aufgabe 6)

Diese Aufgabe soll euch ein Gefühl für RegEx geben.

- lasst euch eure IP-Adresse ausgeben und schreibt die Ausgabe in `ip.txt`
- lasst euch nun nur die Zeilen Ausgeben, die eine valide IP-Adresse enthalten (IPv4, IPv6, oder beides)
- **Achtung!!** es gibt mehrere Lösungen

Tipp: `ip addr`, `grep`, `RegEx`