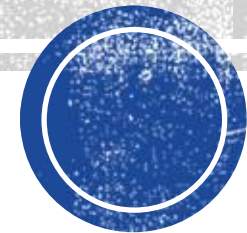


LINUX TERMINAL

Tutorial für Anfänger

von 10:00 - 16:00 Uhr



Ein Angebot des Fachbereichs Informatik in Kooperation mit dem Stud. Trainerpool

h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
STUDENTISCHER TRAINERPOOL

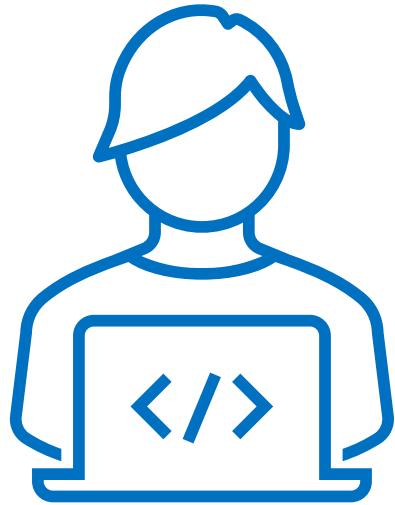


SCAN ME

Seminarangebot:

Sozialkompetenzen	Selbstkompetenzen	Methodenkompetenzen
Kommunikation	Stressbewältigung	Excel & Word Kurse
Konfliktdynamiken	Motivation	Python & LaTeX Kurse
Teamleading	Lernstrategien	Projektmanagement

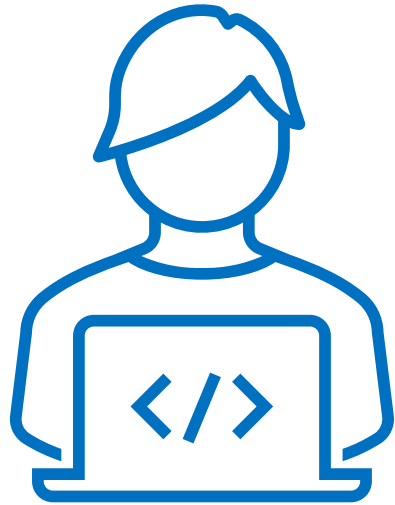
- viele weitere Seminare
- Anmeldung online
- Angebote in Präsenz und online
- Vorlesungszeit: Samstags (Präsenz)
- Semesterferien: unter der Woche



INHALTSVERZEICHNIS

- 1. Einstieg
- 2. Navigation in command line
- 3. Dateien und Verzeichnisse
- 4. Dateien mit Vim bearbeiten
- 5. Ströme und Umleitung
- 6. Subshells
- 7. Prozessmanagement
- 8. Installieren mit apt
- 9. Automatisieren mit Script
- Feedback





WIFI-ZUGANG

SSID: **events.hda**

Passwort: **22/event/10**



1. EINSTIEG: COMMAND LINE

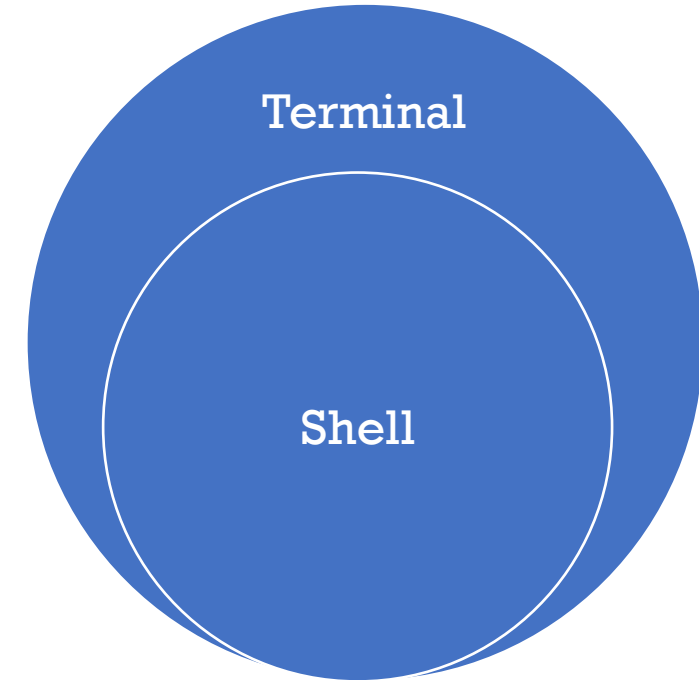
- Was ist eine Command line (terminal, console etc.)?
 - textbasiertes Tool zur Steuerung des Computers

Command line	GUI (Graphical User Interface)
<p>Pro:</p> <ul style="list-style-type: none"> • verfügbar auf jedem Computer • einfache Entwicklung von Software(Command line interface) • Aufgabenautomatisierung via Skripts – Weniger Knöpfe klicken <p>Kontra:</p> <ul style="list-style-type: none"> • anfänglich unintuitive Benutzerführung 	<p>Kontra:</p> <ul style="list-style-type: none"> • nicht immer auf gewünschter Plattform verfügbar (z.B. Server) • benötigt mehr Ressourcen • zusätzlicher Entwicklungsaufwand für GUI-Design • benötigt menschlichen Nutzer zur Bedienung <p>Pro:</p> <ul style="list-style-type: none"> • intuitive Benutzerführung



1. EINSTIEG: WARUM LINUX?

- Terminal (commandline interface)
 - Stellt ein Interface für textbasierte Ein-/Ausgabe bereit
- Shell (Interpretationsprogramm)
 - geladen beim Start vom Terminal
 - interpretiert Nutzereingaben im Terminal
 - Ausgabe von Befehlen im Terminal
- Konsole (physisches Objekt)
 - Computer mit Tastatur, welcher Terminal ausführt



1. EINSTIEG: WARUM LINUX?

- kostenlos
- open source (mitunter Unabhängigkeit von Firmen*)
- performanter (d.h. schneller als z.B. Windows)
- robustere Sicherheitsstrukturen
- volle Kontrolle über Systemupdates
- gute Softwareverfügbarkeit

*abhängig von Distribution



1. EINSTIEG: WARUM LINUX?

- **Docker Image zum ersten Mal laden:**

```
docker run --name ubuntu-playground-c -it jonaswessner/ubuntu-playground:latest
```

- **Dockercontainer fortsetzen nach dem Verlassen:**

```
docker start -i ubuntu-playground-c
```

- **Dockercontainer und seine Daten/Zustand entfernen:**

```
docker container rm ubuntu-playground-c
```





FRAGEN?

Welche Fragen habt ihr zum
Thema 1. Einstieg?



2. NAVIGATION IN COMMAND LINE

- aktuelle Position von Shell: **pwd** (Print Working Directory)
- Position beim Start: home directory vom Nutzer

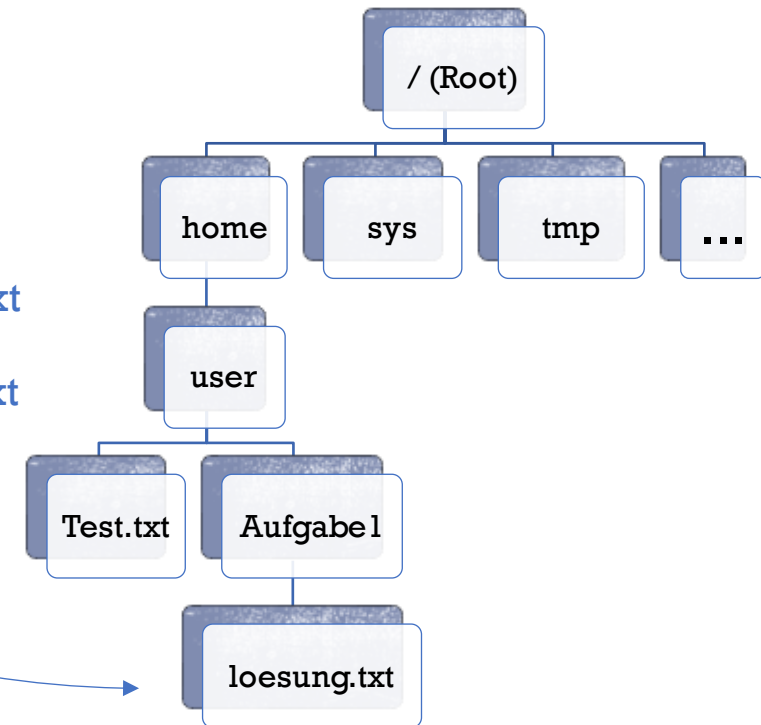
```
[myuser@99ac344ce626:~$ pwd  
/home/myuser
```

- user type „root“: Nutzer mit privilegiertem Zugang zum System



2. NAVIGATION: LINUX-DATEISYSTEM

- Organisiert als Baum:
 - / (Root): Stammverzeichnis
 - Position angegeben als „Pfad“
 - (absolut, start: Root) Bsp. : `/home/user/Aufgabe1/loesung.txt`
 - (relativ, start: working directory) Bsp.: `./Aufgabe1/loesung.txt`
oder `Aufgabe1/loesung.txt`
 - ein Verzeichnis höher (Richtung Root): `..`
 - `~` ist äquivalent zu `/home/user`
- Verzeichnis wechseln (change directory): `cd <Zielpfad>`



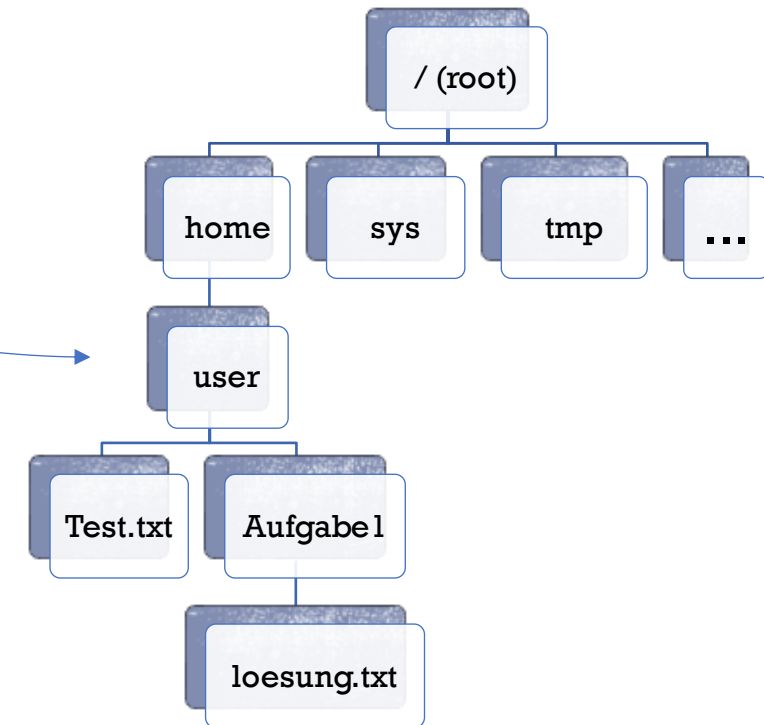
2. NAVIGATION: LINUX-DATEISYSTEM

Auflistung von Inhalten:

- **ls**
Test.txt Aufgab1
- **ls** <option> Details in Kapitel 3

Tipps:

- Tab-Taste für Autovervollständigung
- Pfeiltasten ruft vorherige Befehle auf
- **history** erzeugt Liste verwendeter Befehle





FRAGEN?

Welche Fragen habt ihr zum
Thema 2. Navigation?





AUFGABE 2: NAVIGATION

- Aufgaben unter [home/myuser/2_navigation/task.txt](#)
- Lösung unter [home/myuser/2_navigation/solution.txt](#)

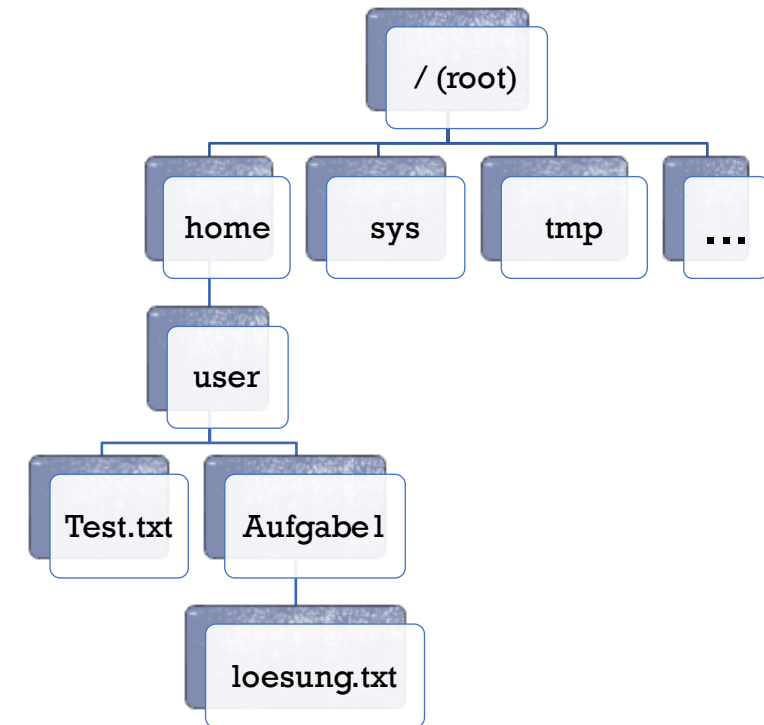
```
myuser@94c3ae74e18c:~$ ls
2_navigation      4_vim              6_subshells        8_apt              README             reset.sh
3_files_and_dirs  5_pipes_redirection_search  7_process_management  9_scripting        backup
myuser@94c3ae74e18c:~$ cd 2_navigation/
myuser@94c3ae74e18c:~/2_navigation$ ls
docs  photos  solution.txt  task.txt
myuser@94c3ae74e18c:~/2_navigation$
```

1. Show all contents of this directory
2. Find the file called 'passwort.secret' and look at its content.
3. Navigate to the root of the directory tree.
4. Show all home directories on the machine
5. Show the contents of the directory /bin. What do you see?
6. Change to your home directory using the '~' syntax
7. Navigate to locations of your own choice using absolute and relative paths.
8. Use the command history (arrow up and down) to execute some of the commands, which you have used previously once again.
9. Print your current location in the directory tree



3. DATEIEN UND VERZEICHNISSE

- **ls -l** listet Inhalt des Verzeichnisses auf (vertikal)
 - Test.txt
 - Aufgabe1
- **ls -l** listet Inhalt des Verzeichnisses auf (detailliert)
 - `-rw-rw-r-- 1 user user 0 Oct 10 10:22 Test.txt`
 - `dwxr-xr-x 2 user user 4096 Oct 10 10:22 Aufgabe1`
 - **directory**, **r**ead, **w**rite, **e**xecute
 - [Typ] [Zugriffsrecht: Nutzer(Besitzer), Gruppe(Besitzer), Andere] [Hard link] [Nutzer] [Gruppe] [Größe] [Datum]
[Name]



3. DATEIEN UND VERZEICHNISSE: ZUGRIFFSRECHTE

- Befehl **chmod** <opt> <Dateiname> verwaltet Zugriffsrechte
 - **opt** spezifiziert welche Rechte geändert werden
 - **+x** gibt allen Nutzern & Gruppen Ausführungsrecht
 - **-x** entzieht allen Nutzern & Gruppen Ausführungsrecht
 - **u+x** gibt dem Nutzer(Besitzer) Ausführungsrechte
 - **u-x** entzieht dem Nutzer (Besitzer) Ausführungsrechte
 - analog für Gruppen (**g**roup), andere Nutzer (**o**thers)
 - analog für andere Rechte (**r**ead, **w**rite)



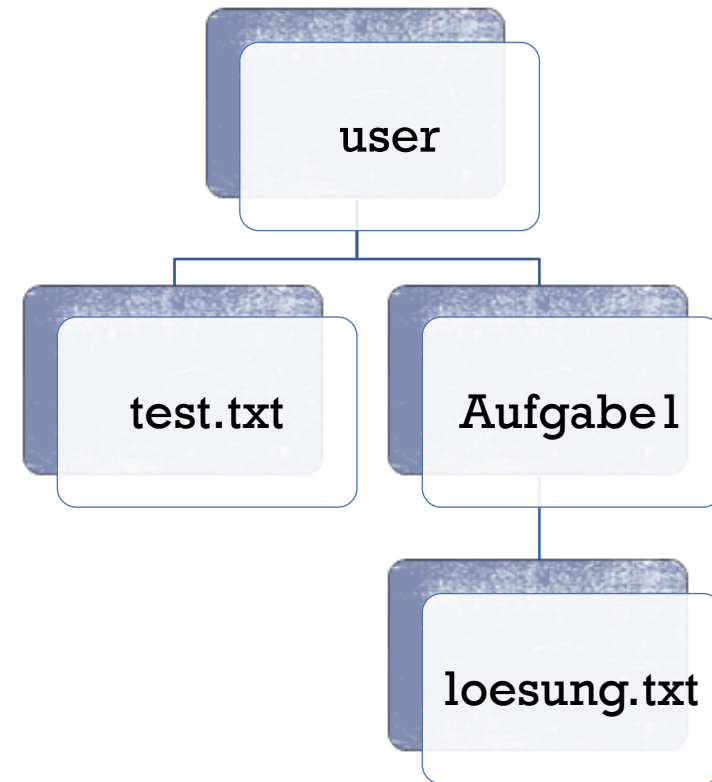
3. DATEIEN UND VERZEICHNISSE

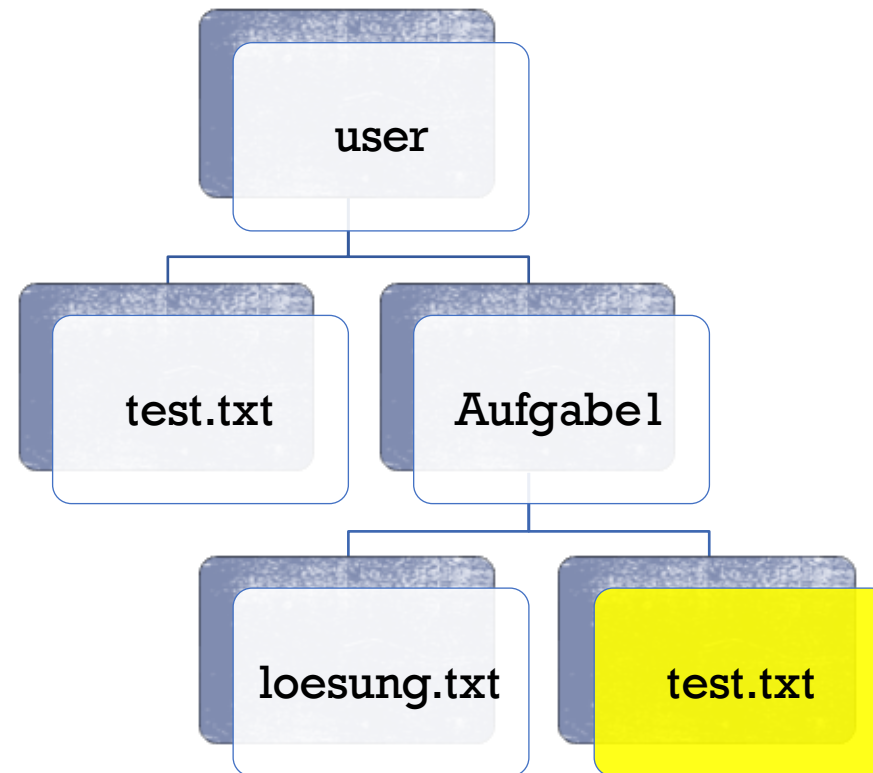
- Datei-/Verzeichnispfade sind einzigartig
- Dateiendung hat keinen Einfluss auf Inhalt
- Dateien startend mit . sind verborgen (siehe ls -A)
- **cat** <Dateiname> zeigt Dateiinhalt an
- **touch** <Dateiname> erstellt leere Datei
- **mkdir** <Verzeichnisname> erstellt Verzeichnis



3. DATEIEN UND VERZEICHNISSE

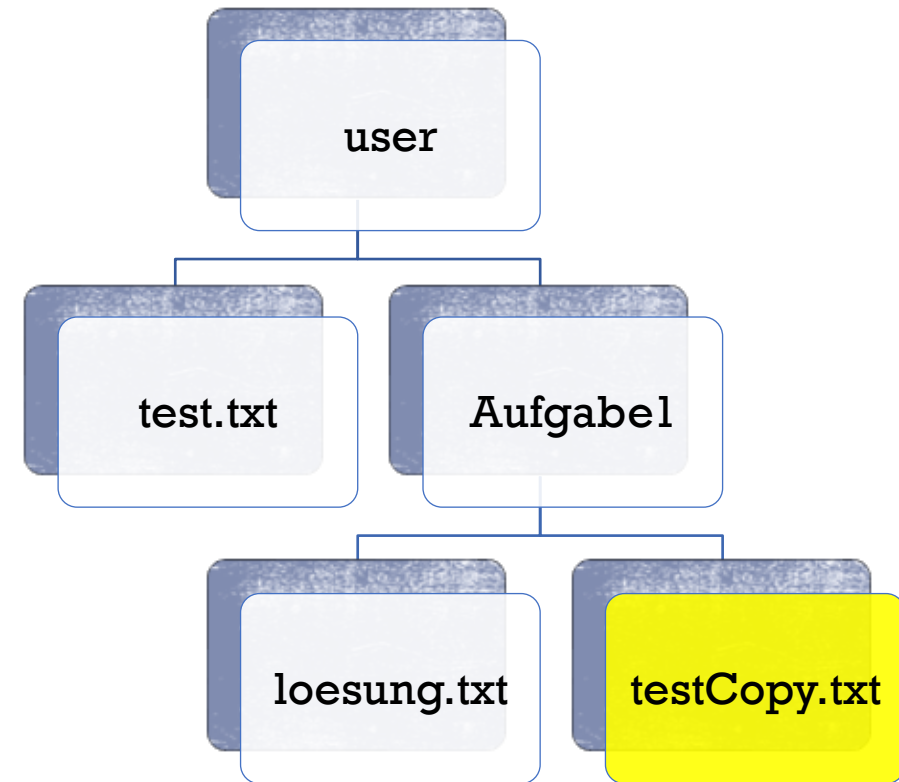
- **rm** <Dateiname> löscht Datei
- **rm -r** <Verzeichnisname> löscht Verzeichnis
- **cp** <Quellpfad> <Zielpfad> kopiert Datei
 - Bsp. (start bei user): **cp** test.txt Aufgabe1





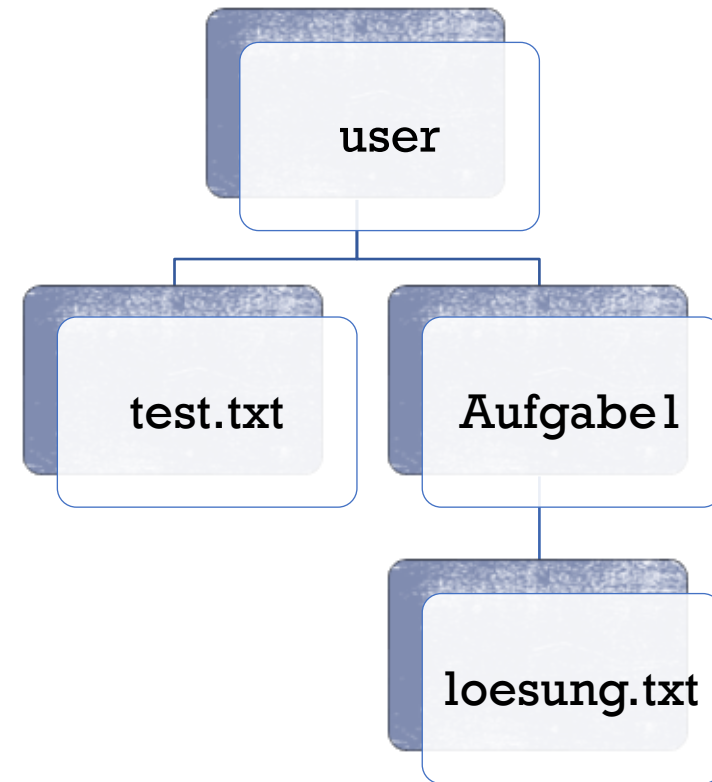
3. DATEIEN UND VERZEICHNISSE

- Bsp.: **cp** test.txt **Aufgabe1/testCopy.txt**
 - => Kopieren und Umbenennen



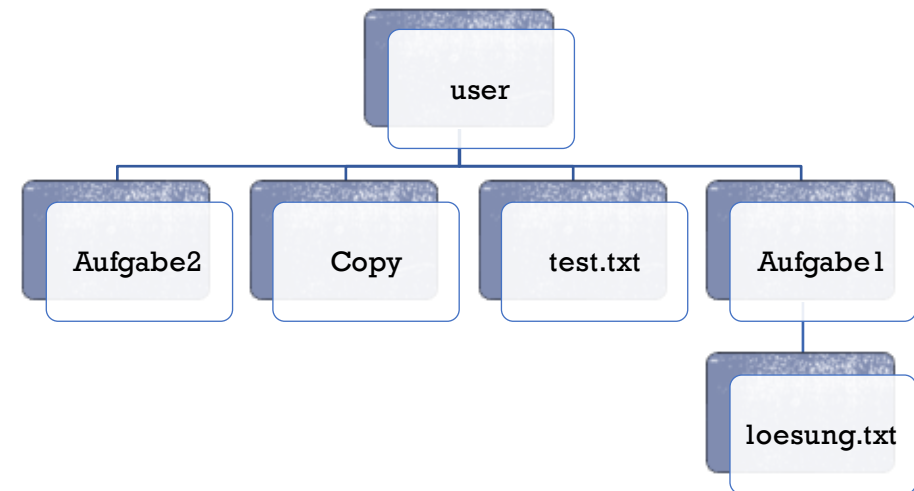
3. DATEIEN UND VERZEICHNISSE

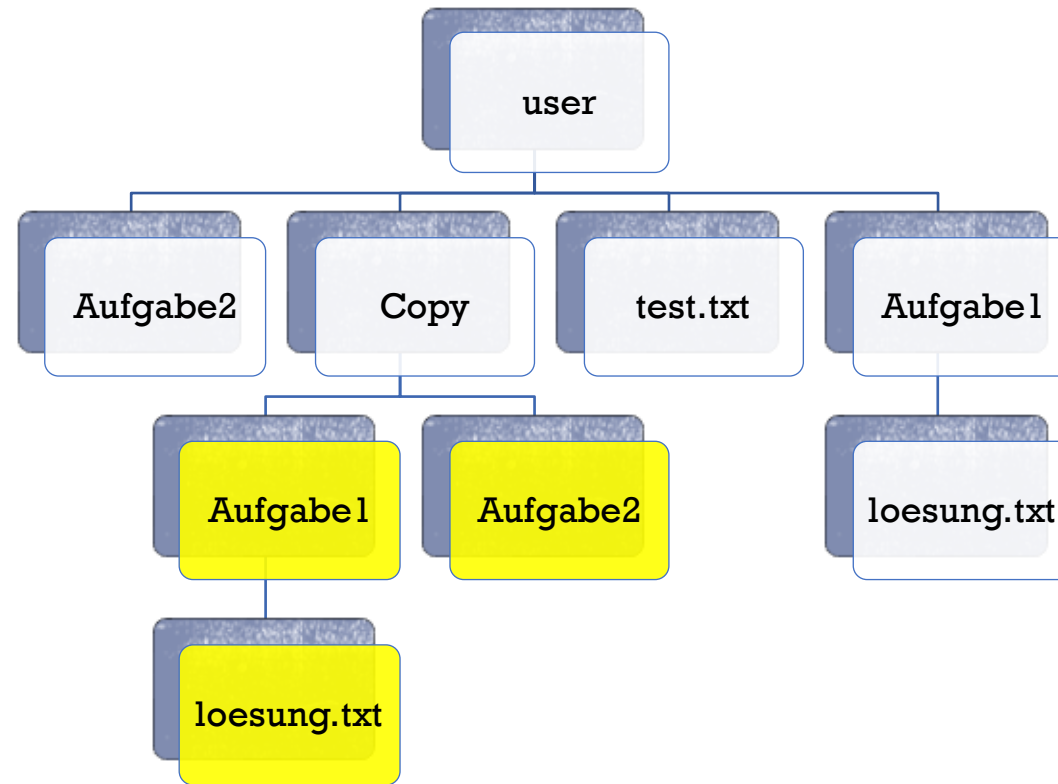
- Überschreiben existierender Dateien möglich!
 - `cp test.txt Aufgabe1/loesung.txt`
- `cp -r <Quellpfad> <Zielpfad>` kopiert Verzeichnis
- analog: `mv <Quellpfad> <Zielpfad>`
verschiebt Datei/Verzeichnis; siehe Umbenennung



3. DATEIEN UND VERZEICHNISSE

- Auswahl von Dateien mit ähnlichem Bezeichnungsschema => Wildcard verwenden
- Bsp. `cp -r A* Copy`





Alle Dateien/Verzeichnisse beginnend mit A wurden ins Verzeichnis Copy kopiert!





FRAGEN?

Welche Fragen habt ihr zum
Thema 3. Dateien und Verzeichnisse?





AUFGABE 3:

DATEIEN UND VERZEICHNISSE

```
[myuser@94c3ae74e18c:~/3_files_and_dirs$ cat task.txt
1. Show all contents of this directory
2. Show all contents of this directory in separate lines including hidden files
3. Show all contents of this directory with their permissions
4. Create a new file called `my_file.txt`
5. Delete the file called `useless.json`
6. Create a directory with the name `new_files`
7. Move all files with the suffix `.new` to the directory `new_files`
8. Remove all files with the suffix `.old?`
9. Rename the file `weird_name.txt` to `good_name.txt`
10. Clear all contents of the directory `trash`
11. Use the command `./check.sh` in this directory to verify your results
```


NOTE 1: Wildcards (*) can be helpful

NOTE 2: Watch out for hidden files



4. DATEIEN BEARBEITEN MIT VIM

- Vim (command line editor) auf vielen Linux Distros vorhanden
- **vim** <Dateiname> öffnet Datei in vim
- Verschiedene Modi:
 - Command: Initialmodus, lx esc zum Verlassen eines Modus
 - Insert Mode (Taste **i**); Bearbeitungsmodus
 - 2x **d** tippen um ganze Zeile zu löschen
 - **:q** eingeben zum Verlassen von vim (wenn keine Änderung)
 - **:q!** eingeben zum Verlassen von vim ohne Speichern
 - **:wq** eingeben zum verlassen von vim mit Speichern



```
Test
Test2
Test3
Test4
~
~
~
~
~
~
~
~
-- INSERT --      4,5      All
```





FRAGEN?

Welche Fragen habt ihr zum
Thema 4. Dateien mit Vim bearbeiten?





AUFGABE 4:

DATEIEN MIT VIM BEARBEITEN

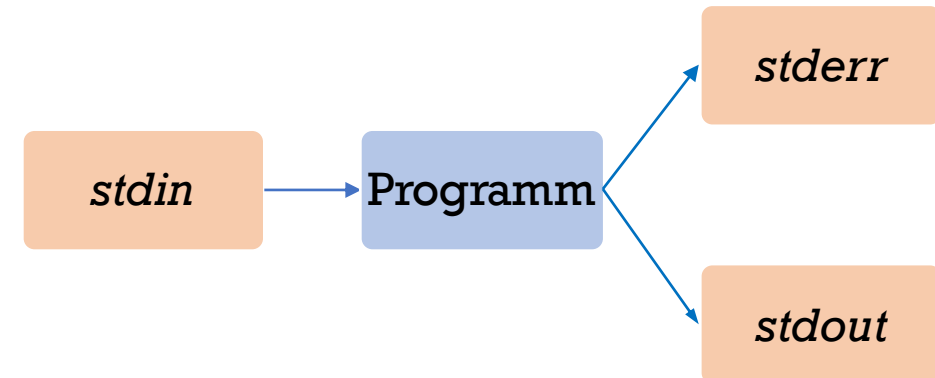
```
myuser@94c3ae74e18c:~/4_vim$ cat task.txt
1. Create a file called 'lets_go_vim.txt'
2. Use vim to edit the file 'lets_go_vim.txt' and change its contents to 'Hello Vim!'
3. Look at the modified file using the cat command
4. Use Vim to replace all numbers with dots in the file 'shopping_list.txt' by hyphens (-).
   E.g. the first line should look like this '- Water'.
5. Delete the entry with cream in the file 'shopping_list.txt'
6. Use the command './check.sh' in this directory to verify your results

NOTE: Do not add any extra white spaces in the files for the check command to work properly.
```



5.STRÖME UND UMLEITUNG

- Programme besitzen 3 Nachrichtenströme
 - Standardeingabestrom *stdin*
 - Standardausgabestrom *stdout*
 - Standardfehlerstrom *stderr*



5.STRÖME UND UMLEITUNG

- Beispiel:

```
myuser@94c3ae74e18c:~/9_scripting$ egrep "aaa"
```

```
Nach enter kommt stdin aaa
```

```
Nach enter kommt stdin aaa
```

Programm

stdin

stdout

```
myuser@99ac344ce626:~/3_cde$ ls ./2_files
```

```
ls: cannot access './2_files': No such file or directory
```

```
myuser@99ac344ce626:~/3_cde$
```

stderr



5.STRÖME UND UMLEITUNG

- Ströme vom Terminal umleiten:
 - `>` leitet stdout in eine Datei um (erstellt/ersetzt Datei)
 - Bsp.: `pwd > verzeichnis.txt`
 - `>>` leitet stdout in eine Datei um (erstellt/ergänzt Datei)
- `<` leitet Inhalt von einer Datei zu stdin um
 - Bsp.: `egrep "l" < datenbank.txt`



5.STRÖME UND UMLEITUNG

- Ströme vom Terminal umleiten:
 - | verbindet stdout von Prgm 1 mit stdin von Prgm 2 (Piping)
 - Bsp: **ls** | **egrep** “.txt“





FRAGEN?

Welche Fragen habt ihr zum
Thema 5. Ströme und Umleitung?





AUFGABE 5:

STRÖME UND UMLEITUNGEN

```
myuser@94c3ae74e18c:~/5_pipes_redirection_search$ cat task.txt
1. Copy the files 'best_recipe.txt.new' and 'morning_routine.txt.new' from exercise
2 to this directory
2. Append the contents of 'best_recipe.txt.new' to the file 'morning_routine.txt.new'
' using output stream redirection
3. Create a file called 'search_results.txt'
4. Search for all lines containing the substring ',Q' in the file 'titanic_train.csv'
' using egrep and append the search result to the file 'search_results.txt'
5. Create a file called 'count.txt'
6. Use the 'egrep' command and the command 'wc -l' to count the number of lines in t
he 'titanic_train.csv' containing the substring ',S'. Write the result to the file '
count.txt'.
7. Use the command './check.sh' in this directory to verify your results

NOTE: The 'wc' command reads from its standard input stream
```



6. SUBSHELLS

- Ausführen von Befehlen innerhalb eines Befehls
 - verschachtelte Ausführung von Befehlen/Programmen mit **\$(subcommand)**
 - Bsp.: **echo** "Your current directory is **\$(pwd)**" =>

```
myuser@99ac344ce626:~/3_cde$ echo "Your current directory is $(pwd)"  
Your current directory is /home/myuser/3_cde
```

- Ausführung von innen nach außen
- Quiz: **mkdir** **\$(pwd)**/**new_dir**





FRAGEN?

Welche Fragen habt ihr zum
Thema 6. Subshells?





AUFGABE 6: SUBSHELLS

```
myuser@94c3ae74e18c:~/6_subshells$ cat task.txt
1. Use the echo command, a subshell and the date command to print the text 'It is now <date>',
[ where date is the current date and time. Redirect the output to a file called 'date.txt' ]
2. Use a subshell and the 'date +%y_%m_%d_%H' command to automatically create a file with the ]
[ name '<year>_<month>_<day>_<hour>.log', where the in brackets encapsulated variables equal the ]
  attributes of the current point in time.
3. Check whether the entries with the ID 121 are equal for the file 'titanic_train_1.csv' and ]
  the file 'titanic_train_2.csv' using the egrep command and a subshell
4. Repeat the third task for the id 122
5. Use the command './check.sh' in this directory to verify your results for task 1 and 2
6. Compare your results for exercise 3 and 4 with those of the person next to you or check the
  file 'solution.txt'
```



7. PROZESSMANAGEMENT

- Prozess = Instanz eines ausführbaren Programms (z.B. mehrere Firefox-Fenster)
 - voneinander unabhängig
 - gleiches Programm (identisches Programmverhalten)
- Im Terminal laufen Programme als Vorder-/ bzw. Hintergrundprozess
 - Vordergrundprozesse blockieren Shell
 - Hintergrundprozesse laufen ohne Shell zu blockieren (geeignet z.B. für Programminstallation)



7. PROZESSMANAGEMENT: HINTERGRUNDPROZESS

- **&** nach Befehl führt diesen im Hintergrund aus
- Bsp.: **sleep 60 &**

```
myuser@99ac344ce626:~/3_cde$ sleep 60 &  
[1] 325  
myuser@99ac344ce626:~/3_cde$ jobs  
[1]+  Running                  sleep 60 &  
myuser@99ac344ce626:~/3_cde$
```

- Status von Hintergrundprozessen mit **jobs** anzeigen



7. PROZESSMANAGEMENT: HINTERGRUNDPROZESS

- **fg** <jobID> bewegt Hintergrundprozess in den Vordergrund

```
[myuser@99ac344ce626:~/3_cde$ jobs  
[1]+  Running                  sleep 60 &  
[myuser@99ac344ce626:~/3_cde$ fg 1  
sleep 60  
█
```

- Ctrl + Z pausiert Vordergrundprozess

```
^Z  
[1]+  Stopped                  sleep 60  
myuser@99ac344ce626:~/3_cde$ █
```

Unter Windows
funktioniert Ctrl + Z nicht.
(Bug von Docker)



7. PROZESSMANAGEMENT: HINTERGRUNDPROZESS

- **fg** <jobID> startet pausierten Prozess wieder im Vordergrund
- **bg** <jobID> startet pausierten Prozess wieder im Hintergrund
- Ctrl + C bricht Vordergrundprozess ab





FRAGEN?

Welche Fragen habt ihr zum
Thema 7. Prozessmanagement?





AUFGABE 7: PROZESSMANAGEMENT

```
myuser@94c3ae74e18c:~/7_process_management$ cat task.txt
1. execute the command '$(sleep 180 ; echo "finished at $(date)" > finished.txt)' as a
background process
2. Save the current output of the command 'jobs' to a file called 'running.txt'
3. Pull the process started earlier into the foreground
4. Stop the process (Do not cancel it but stop it temporarily!
5. Save the current output of the command 'jobs' to a file called 'stopped.txt'
6. Resume the process in the background
7. Wait until the background process has finished and try to understand what happened.
8. Use the command './check.sh' in this directory to verify your results for tasks 1-6
9. Discuss your results of task 7 with the person next to you or verify your hypothesis
using the file 'solutions.txt'
```



8. INSTALLIEREN MIT APT

- Vorinstallierte Programme sind z.B. **ls**, **pwd**, **egrep** etc.
- **apt** ermöglicht Installation weiterer Programme
 - Paketmanager
 - verwaltet Liste aller installierbaren/installierten Programme auf dem Computer
 - **apt update** aktualisiert Liste auf neuste Version (empfohlen!)
 - **apt upgrade** updatet alle installierten Programme
 - **apt install** <package> sucht und installiert besagtes Paket/ Programm
 - **apt remove** <package> deinstalliert besagtes Paket/Programm
- apt benötigt **sudo** Befehl (SuperUserDo)
- ermöglicht automatisierte Installation mehrerer Programme
- nicht alle Programme verfügbar





FRAGEN?

Welche Fragen habt ihr zum
Thema 8. Installieren mit apt?





AUFGABE 8:

INSTALLIEREN MIT APT

```
myuser@94c3ae74e18c:~$ cd 8_apt/  
myuser@94c3ae74e18c:~/8_apt$ cat task.txt  
1. Install the program 'unzip' using apt  
2. Unzip the zip-archive 'big_data.zip' using the program unzip  
3. Uninstall the program 'unzip' using apt  
4. Use the command './check.sh' in this directory to verify your results  
  
NOTE 1: Installing and uninstalling requires administrator rights (sudo)  
NOTE 2: The password for the super user (sudo) ist 'secret'
```



9. AUTOMATISIERUNG MIT SCRIPT

- Script (Skript)
 - Programme bestehen aus mehreren Befehlen (siehe vorh. Kapitel)
 - Programm wird während des Ausführens kompiliert (JIT)
 - Bash stellt Programmiersprache zur Verfügung (inklusive Schleifen, if & else etc.)
 - Skripte ermöglichen Automatisierung repetitiver Aufgaben
 - Skripte ausführbar nur mit execute Recht



9. AUTOMATISIERUNG MIT SCRIPT

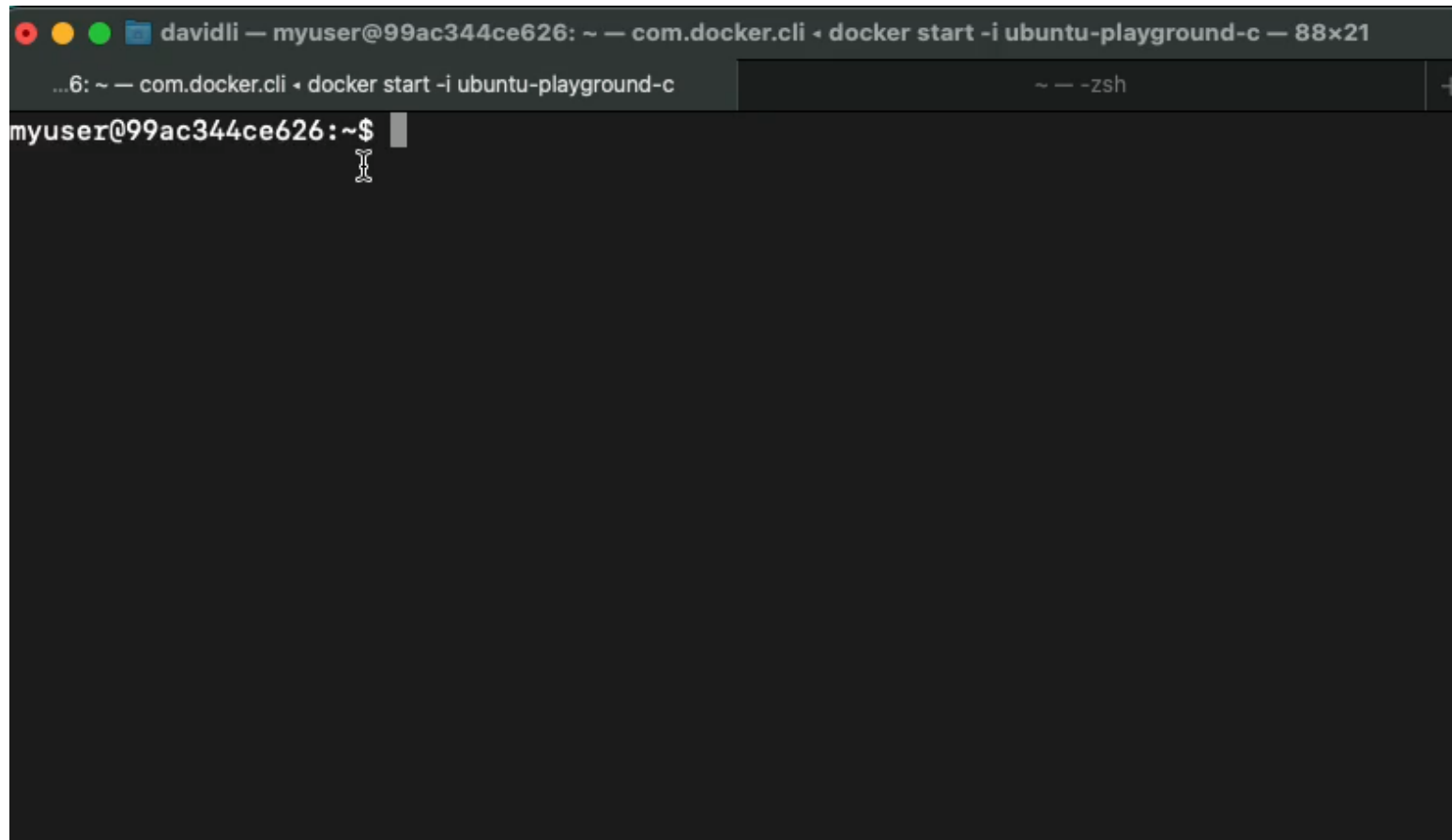
```
#!/bin/bash

echo "This program creates a new directory in home directory called linuxBestOS. A log file is created."
read -p "Press enter to continue"
cd ~
mkdir linuxBestOS && echo "directory created"
cd linuxBestOS
touch willkommen.txt && echo "Ausfuerungszeit ist $(date). Aktuelles Verzeichnis ist $(pwd)" > willkommen.txt
sleep 2 && echo "Abgeschlossen"
cat willkommen.tx
```

"skript.sh" 11L, 381C 11,18 All



9. AUTOMATISIERUNG MIT SCRIPT



A terminal window with a dark background and light text. The title bar at the top shows window control buttons (red, yellow, green) and the text "davidli — myuser@99ac344ce626: ~ — com.docker.cli • docker start -i ubuntu-playground-c — 88x21". Below the title bar, there are two tabs: the first is "...6: ~ — com.docker.cli • docker start -i ubuntu-playground-c" and the second is "~ — -zsh". The main area of the terminal shows the prompt "myuser@99ac344ce626:~\$" followed by a cursor and a small grey rectangle.





FRAGEN?

Welche Fragen habt ihr zum
Thema 9. Automatisierung mit Script?





AUFGABE 9: INSTALLIEREN MIT APT

```
myuser@94c3ae74e18c:~/9_scripting$ cat task.txt
This directory is an exact copy of the directory '2_files_and_dirs'.
This time it is your task to create a script, which completes the exercise for you.
As a reminder, the tasks of exercise 2, which should be automated in this exercise, are:

- Show all contents of this directory
- Show all contents of this directory in separate lines including hidden files
- Show all contents of this directory with their permissions
- Create a new file called `my_file.txt`
- Delete the file called `useless.json`
- Create a directory with the name `new_files`
- Move all files with the suffix `.new` to the directory `new_files`
- Remove all files with the suffix `.old?`
- Rename the file `weird_name.txt` to `good_name.txt`
- Clear all contents of the directory `trash`

Bonus: If that is fun, implement another script for one of the other exercises.

NOTE: If you need to reset the directory to test your script multiple times, then use the './reset 8_scripting'
command in your home directory. If you do that you might want to backup your script somewhere outside of this
directory, so it will not get deleted during the reset process.myuser@94c3ae74e18c:myuser@94c3ae74e18c:myusemyu
```





FEEDBACK

Wie hat euch unser Kurs gefallen?

Wurden eure Erwartungen erfüllt/habt ihr Verbesserungsvorschläge?





FEEDBACK

