# Einführung in die Informatik

## 04 Bedienung eines klassischen Betriebssystems

Prof. Dr. Carsten Link

Stand: 6. November 2024

## Inhaltsverzeichnis

# 1 Kompetenzen und Lernegebnisse

Durch das Bearbeiten dieses Materialpaketes erwerben Sie diese Kompetenzen (Wissen, Fähigkeiten, Fertigkeiten zur Problemlösung):
**Sie können ein einfaches klassisches Betriebssystem grundlegend bedienen.**
Die oben genannten Kompetenzen erwerben Sie, indem Sie Lernziele erreichen, welche sich prüfen lassen. Lernegebnisse: Sie können nachweislich[1]:

- Eigenschaften BS erläutern
- Eigenschaften verschiedener Nutzungsschnittstellen erläutern
- verschiedene REPLs bedienen
  - Shell: progs starten, stoppen,
  - auf einem entfernten Rechner Shell benutzen
  - Daten zwischen Rechnern austauschen
- mit Dateien und Ordnern umgehen
  - Dateien anordnen in Ordnern
  - Dateien wiederfinden
  - Dateien und Verzeichnissen verschieben und umbenennen
  - absolute und relative Pfade verwenden
  - Namensplatzhalter verwenden
  - Datei-Metadaten interpretieren
  - Dateien aus Containerdateien extrahieren
  - Containerdateien erstellen
- Dateien betrachten und editieren
  - verschiede textbasierierte Dateiformate erläutern und voneinander abgrenzen
  - Editoren für versch. Formate benutzen
  - Betrachter für versch. Formate benutzen
  - geeigneten Text-Editor auswählen
- Tastenkürzel verwenden, um effizient arbeiten zu können
- eine Zusammenfassung des Inhalts dieses Materialpakets erstellen

[1] Sie können das Erzielen der einzelnen Lernergebnisse beispielsweise bei einem Testat im Praktikum oder einer Aufgabe in der Modulprüfung nachweisen

# 2 Konzepte

## 2.1 Historische Entwicklung

## 2.2 Betriebssysteme

### 2.2.1 Aufgaben und Funktionen von Betriebssystemen

Zweck von Computern und Programmen aus der Sicht des Endnutzers: Datenverarbeitung. Daten liegen auf Platten, USB-Sticks, Programme bearbeiten diese Dateien.
Wesentliche Aufgaben und Funktionen von Betriebssystemen:

- Programme ins RAM bringen und ausführen lassen (Prozesse)
- Programme unterstützen (API, Bibliotheken) und isolieren (Speicherschutz, Security)
- dazu nötig: Ressourcenverwaltung
- dazu nötig: Dateisystem
- vorher nötig: Boot
- Systemprogramme
- Sahnehäubchen: Netzwerk

Fazit:

- Betriebssystem: hilft, Programme geordnet auszuführen, damit Nutzer Dateien bearbeiten können
- kanonisches Betriebssystem UNIX: kernel, libraries, C-Compiler, tools, shell, TCP/IP

### 2.2.2 Nutzungsschnittstellen CLI, TUI, GUI

**Command Line Interface (CLI)**: Das Command Line Interface zeichnet sich durch eine sehr einfache Nutzungsweise aus. Es wird ein einzeiliges textuelles Kommando über die Tastatur abgesetzt, welches dann etwas bewirkt und/oder eine oder mehrere Zeilen ausgibt. Der Umgang mit Kommandos wird in Kapitel 2.4 beschrieben.
**TUI**: Das Textual User Interface ist textbasiert und verwendet den sichtbaren Bereich des Terminals vollständig. Beispiele hierfür sind: `nano`, `top`. Die Bedienung erfolgt über die Tastatur. Oft sind zu Buchstaben auch Modifiziertasten (Shift, Control, etc.) gedrückt zu halten.
**GUI**: Die Bedienung eine Graphical User Interfaces erfolgt über Maus oder Touchpad. Die Steuerung per Tastatur ist meist auch möglich.
**Touch UI**: Die Bedienung eine Graphical User Interfaces erfolgt über Gesten direkt auf dem Bildschirm.

## 2.3 Read-Eval-Print-Loop (REPL)

Die Abkürzung REPL steht für Read Eval Print Loop. Damit ist gemeint, dass ein Gerät oder ein Programm Nutzereingaben einliest, diese bearbeitet, die Antwort ausgibt und wieder von vorne anfängt. Ein Beispiel für ein Gerät mit REPL ist ein Taschenrechner. Auf UNIX-Systemen ist mit `bc` einfacher Taschenrechner installiert, der über eine REPL verfügt:

bc -l

```
>>> 10 + 20
30
>>> 3.3 + 4.4
7.7
>>> a = 5
>>> b = 7
>>> 3 * a + 7 + b
29
>>>
```

Die drei Zeichen am Anfang einer Zeile (`>>>`) zeigen an, dass `bc` auf eine Eingabe wartet. Diese Eingabezeile wird bei REPLs Prompt genannt.
Manche REPLs erlauben einfaches Editieren der Eingabe am Prompt[2].
Die Python REPL erlaubt es auch, Funktionen zu definieren und zu nutzen:

python3

```
Python 3.12.3 Type "help", "copyright", "credits" or "license" for more information.
>>> 10 + 20
30
>>> print(1 + 2)
3
>>> print(3.3 + 4.4)
7.7
>>> a = 5
>>> b = 7
>>> print(3 * a + 7 + b)
29
>>> def dup(x):
...     return x * 2
...
>>> print(dup(99))
198
```

Sollen weitere mathematische Operationen verfügbar sein, müssen diese mit `import math` geladen werden. Dann stehen beispielsweise bereit: `sin()`, `pow()`, `log10()`. Eine Liste wird angezeigt mit `math.` TAB TAB.
Viele Programme und vor allem Programmiersprachen verfügen über eine REPL; beispielsweise Java, Julia, Groovy, alle Shells (siehe Kapitel 2.4).

## 2.4 Shell

Ein Programm, das es ermöglicht mit einer REPL mit einem System zu interagieren, werden Shell genannt. Die wichtigste Interaktion hier ist das Starten

[2] Verschiedene Taschenrechner unterscheiden sich hier auch deutlich.

von Programmen. Auf UNIX-Systemen existieren viele Programme, die eine kleine Aufgabe erledigen und sich danach sofort wieder beenden; diese werden Kommandos genannt. Die häufigste eingesetzte Shell ist die Bourne Again Shell (`bash`).
Die übliche Weise, Kommandos zu verwenden hat diese Struktur: `cmd [flags] [short-options] [long-options] [args]]` Parameter in eckigen Klammern sind optional. Am Beispiel von `ls` (GNU) werden die Parameter illustriert:

- `cmd`: `ls`
- `flags`: z.B. `-l` für langes Format oder `-t` Sortierung nach Zeitstempeln
- `short-options`: `-I '*.txt'`
- `long-options`: `--help` oder `--ignore='*.txt'`
- `args`: ein oder mehrere Namen von Verzeichnissen oder Dateien

In der Shell sind folgende Kommandos hilfreich:

- Terminal starten: Rechtsklick auf Verzeichnis oder Desktop; dann „Terminal hier öffnen“
- anzeigen des aktuellen Verzeichnisses (working directory): `pwd`
- aktuelles Verzeichnis wechseln: `cd <dir>`
- Textdatei editieren: `pluma <file>`
- Textdatei editieren: `pluma <file> &` (Terminal nicht blockiert)
- Programm im aktuellen Verzeichnis starten: `./<prog>`
- Verzeichnis erzeugen: `mkdir <name>`
- Datei oder Verzeichnis verschieben/umbenennen: `mv <source> <target>`
- Datei kopieren: `cp <source> <target>`
- Verzeichnis kopieren: `cp -r <source> <target>`
- Datei(en) löschen: `rm <name> ... <name>` “
- Unterschiede zwischen Textdateien anzeigen: `diff <file1> <file2>` oder `meld <file1> <file2>`
- `.tar`-Datei in aktuelles Verzeichnis entpacken: `tar xf <name>`
- `.tar.gz`-Datei oder `.tgz`-Datei entpacken: `tar xzf <name>`
- Textdatei erzeugen: `touch <name.txt>`
- Verzeichnis löschen: `rmdir <name>`
- Programme installieren: `sudo apt install <name>` (z. B. `name = tree` oder `gedit`)
- Hex Dump einer Datei: `hexdump -vC <name>`
- die Datei `name` ausführbar machen: `chmod +x name`

Viele CLI-Programme verfügen über die hilfreichen Optionen `--version`, `-h` und `--help`.

## 2.5 Dateien und Verzeichnisse

Dateisysteme zeichnen sich hierdurch aus:

- Dauerhafte Speicherung von Byteketten (Dateien) mitsamt Metadaten
- Verzeichnisbäume als ad-hoc Lösung zum späteren Wiederauffinden von Dateien
- Manipulation des Verzeichnisbaumes durch Programme und Kommandos

### 2.5.1 Verzeichnisbäume, mkdir, tree

Im Folgenden wird eine kleine Verzeichnishierarchie angelegt, deren Wurzel das Verzeichnis `project_1` ist.

Shell

```
bash$ mkdir project_1
bash$ mkdir project_1/doc
bash$ mkdir -p project_1/src/include
bash$ mkdir project_1/src/cpp
bash$ tree project_1/
project_1/
├── doc
└── src
    ├── cpp
    └── include
```

### 2.5.2 cd, pwd

Im Folgenden wird mit `cd` das aktuelle Verzeichnis gewechselt (working directory, kann mit `pwd` angezeigt werden):

Shell

```
bash$ pwd
/home/devel
bash$ cd project_1/
bash$ cd src/
bash$ cd cpp/
bash$ pwd
/home/devel/project_1/src/cpp
bash$ cd ../../doc/
bash$ pwd
/home/devel/project_1/doc
```

Bedienung von `cd`:

- `cd path` ⇒ wechselt in das angegebene Verzeichnis, welches absolut oder relativ gegeben sein kann sowie `..` enthalten kann
- `cd` ⇒ wechselt in das Heimverzeichnis
- `cd -` ⇒ wechselt in das vorherig besuchte Verzeichnis

### 2.5.3 mv

Im Folgenden wird mit `mv` eine Datei vom einem Verzeichnis in ein anderes Verzeichnis verschoben:

Shell

```
bash$ echo "int main(){return 0;}" > project_1/doc/main.cpp
bash$ tree project_1/
project_1/
├── doc
│   └── main.cpp
└── src
    ├── cpp
    └── include
bash$ mv project_1/doc/main.cpp project_1/src/cpp/
bash$ tree project_1/
project_1/
├── doc
└── src
    ├── cpp
    │   └── main.cpp
    └── include
```

### 2.5.4 Kurzpfade

Besondere Verzeichnisname (Kurzpfade):

- `.` ⇒ current working directory (siehe `pwd`)
- `..` ⇒ directory above
- `~` ⇒ home directory

Im Folgenden ist zu sehen, dass der Kurzpfad `~` beispielsweise für `cd` verwendet werden kann und von der Shell durch einen reguläre Pfad ersetzt wird:

Shell

```
bash$ cd project_1/src/cpp/
bash$ pwd
/home/devel/project_1/src/cpp
bash$ cd ~
bash$ pwd
/home/devel
bash$ echo ~
/home/devel
```

### 2.5.5 Eigenschaften von Dateien

Eine Datei besteht aus Inhalt damit verbundenen Metadaten. Der Inhalt ist eine Folge von Bytes (Anzahl: Null bis zu einer systemabhängigen Grenze). Hier ist eine aus 20 Bytes bestehende Datei gegeben, wobei jedes Byte durch die acht darin enthaltenen Bits dargestellt ist[3]:

[3] `xxd -b hello.txt | cut -d  -f2-8`

hello.txt

```
01001000 01100101 01101100 01101100 01101111 00101100
00100000 01110111 01101111 01110010 01101100 01100100
00100001 00001010 00110000 00110001 00110010 00110011
00110100 00001010
```

Dieselbe Datei lässt sich auch in der häufig verwendeten Form eines *Hex Dumps* darstellen (die lange Zahl links ist die Startaddresse der Zeile):

Hex Dump

```
bash$ hexdump -vC hello.txt
00000000  48 65 6c 6c 6f 2c 20 77  6f 72 6c 64 21 0a 30 31  |Hello, world!.01|
00000010  32 33 34 0a                                       |234.|
```

Zu guter Letzt läßt sich die Beispieldatei auch als Text darstellen (beispielsweise mit `less hello.txt` oder `cat hello.txt`):

hello.txt als Text

```
Hello, world!
01234
```

Die drei verschiedenen Darstellungsarten verdeutlichen, dass sich die Bytefolge in einer Datei auf verschiedene Arten interpretieren läßt. Beispielsweise wird das Hexadezimale Byte mit dem Wert `0a` von Texteditoren als Zeilenwechsel interpretiert – in den beiden anderen Darstellungen nicht.

### 2.5.6 Datei-Metadaten

Datei-Metadaten anzeigen:

ls -l

```
bash$ ls -l hello.txt
-rw-r--r--  1 clink  staff  20 Mar 18 12:45 hello.txt
```

Metadaten ändern (Zeitstempel):

touch

```
bash$ ls -l test.txt
-rw-r--r--  1 devel  devel  123 Mar 26 10:10 test.txt
bash$ touch test.txt
bash$ ls -l test.txt
-rw-r--r--  1 devel  devel  123 Apr  1 05:16 test.txt
```

Metadaten ändern (Name):

rename

```
bash$ ls -l test.txt
-rw-r--r--  1 devel  devel  123 Apr  1 05:16 test.txt
bash$ mv test.txt test-neu.txt
bash$ ls -l test-neu.txt
-rw-r--r--  1 devel  devel  123 Apr  1 05:16 test-neu.txt
```

Der Name `test.txt` wurde in `test-neu.txt` geändert; die Größe `123` und Zeitstempel `Apr 1 05:16` haben sich nicht geändert.
Datei-Metadaten detailliert:

stat

```
bash$ stat hello.txt
84 33749 -rw-r--r-- 1 devel devel 111858 28 "Apr 21 00:28:28 2021"
"Mar 26 12:02:34 2020" "Mar 26 12:02:34 2020" "Mar 26 11:39:58 2020"
32768 8 0 hello.txt
bash$
bash$ stat -s hello.txt
st_dev=84 st_ino=33749 st_mode=0100644 st_nlink=1 st_uid=1001
st_gid=1001 st_rdev=111858 st_size=28 st_atime=1618957708
st_mtime=1585220554 st_ctime=1585220554 st_birthtime=1585219198
st_blksize=32768 st_blocks=8 st_flags=0
```

### 2.5.7 Dateien suchen

Dateien lassen sich mit dem Kommando `find` suchen, welches vielerlei Suchfilter unterstützt. Ein bei der Suche häufig verwendetes Dateiattribut ist der Name:

find

```
bash$ find . -iname "*.txt"
./test.txt
./lines.txt
./err.txt
```

Das oben angegebene Kommando findet alle Dateien im aktuelle Verzeichnis (und darunter), deren Name auf das Muster `*.txt` passt (case insensitive). Wenn Wildcards (`*` oder `*`) verwendet werden, sollte das Suchmuster in Anführungsstriche gesetzt werden, da es sonst passieren kann, dass die Shell die Wildcards expandiert und es zu einer verfälschten Suche kommt.
Das folgend angegebene Kommando findet alle Dateien im aktuelle Verzeichnis (und darunter), deren Modifikationszeitstempel (modification time, mtime) innerhalb des letzten Tages (d.h. 24h) liegt. Ander Zeiträume sind mit einem der Buchstaben `smhdw` für Sekunden, Minuten, Stunden, Tage und Wochen anzugeben.

find

```
bash$ find . -mtime -1d
.
./test.txt
```

Mit diesen Optionen lassen sich Dateien einer bestimmten Größe finden:

find

```
find .  -size +500k -size -10M
```

Mit dem Kommando `grep` lassen sich Dateien finden, die bestimmte Textmuster enthalten. Zunächst werden zwei Dateien angelegt, deren Inhalt im Folgenden bekannt ist:

Create Text File

```
bash$ echo Hallo > hallo.txt
bash$ echo "Zeile ohne Begrüßung" >> hallo.txt
bash$ echo Hello > hello.txt
bash$ echo "Line without greeting" >> hello.txt
```

Nun wird mit dem Suchprogramm `grep` inhaltlich durchsucht:

grep

```
bash$ grep  llo *.txt       # Dateinamen mit Fundstellen
hallo.txt:Hallo
hello.txt:Hello
bash$ grep -h llo *.txt     # nur Fundstellen
Hallo
Hello
bash$ grep -l llo *.txt     # nur Dateinamen
hallo.txt
hello.txt
bash$ grep -hv llo *.txt    # nur Zeilen ohne Treffer
Zeile ohne Begrüßung
Line without greeting
```

Wichtige Optionen für das `grep`-Kommando:

- `-h` ⇒ nur Fundstelle
- `-l` ⇒ nur Dateiname
- `-r` ⇒ recursive
- `-i` ⇒ case insensitive matching
- `-I` ⇒ binärdateien ignorieren
- `-v` ⇒ Selected lines are those not matching any of the specified patterns

### 2.5.8 Containerdateien: tar

Das Werkzeug `tar` wurde ursprünglich entwickelt, um Verzeichnisse mitsamt der darin enthaltenen Dateien auf einem Band zu sichern. Genau wie `vi` gehört es zur Standardausstattung eine UNIX-Systems und sollte bedient werden können. Mit diesem Kommando wird das Heimverzeichnis des Nutzers `/home/bart/` in die Datei `backup01.tar` geschrieben

tar

```
bash$ tar cf backup01.tar -C /home bart
```

Mit diesem Kommando lässt sich der Inhalt der Datei `backup01.tar` anzeigen:

tar

```
bash$ tar tf backup01.tar
```

Mit diesem Kommando lässt sich der Inhalt der Datei `backup01.tar` in das aktuelle Verzeichnis entpacken:

tar

```
bash$ tar xf backup01.tar
```

Wichtige Optionen für das `tar`-Kommando:

- `-f <file name>` ⇒ use `file name` instead of tape
- `-c` ⇒ create

- `-x` ⇒ extract
- `-v` ⇒ verbose; Ausgabe der Aktivtät
- `-z` ⇒ compress or decompress using gzip (`.tgz` or `.tar.gz`)
- `-C <dir>` ⇒ `cd` into `dir` before adding or extracting entires
- `-p` ⇒ preserve permissions

Die `tar`-Implementierung von FreeBSD (`bsdtar`) kann für viele weitere Containerformate verwendet werden (`.zip` etc.). Wird immer `bsdtar` verwendet, müssen nicht verschiedene Kommandos und deren Parameter auswendig gelernt werden.

## 2.6 Remote Shell (SSH) and Remote Copy (SCP)

Die beiden zusammengehörigen Programme `ssh` und `scp` ermöglichen das Arbeiten mit entfernten Rechnern über verschlüsselte Netzwerkverbindungen. Die Syntax ist: `ssh username@remote_host -p <port>` (die Option `-p` ist nur notwendig, wenn von dem Standardport `22` abgewichen wird.)
Hier ein Anmeldevorgang auf einer virtuellen NetBSD-VM:

ssh
```
bash$ uname -a
r8d8.local Darwin Kernel Version 22.6.0: Mon Apr 22 2024; arm64
bash$ ssh -p3022 devel2@localhost
(devel2@localhost) Password for devel2@netbsd:
Last login: Wed May 22 18:08:56 2024 from 10.0.2.2
NetBSD 9.3 (GENERIC) #0: Thu Aug  4 15:30:37 UTC 2022
Welcome to NetBSD!
bash$ uname -a
NetBSD 9.3 (GENERIC) Aug 4 2022 amd64
```

Mit der Option `-X` ist es möglich, die Fenster von grafischen Programmen auf das lokale System zu bringen (X-Forwarding).
Mit dem Kommando `scp` können Dateien und Verzeichnisse kopiert werden.
Die Syntax ist: `scp -P <port> local_file remote_username@remote_host:remote_dir`.
Wird `remote_dir` hinter dem Doppelpunkt weggelassen, so wird in das Heimverzeichnis kopiert. Ein Beispiel:

scp
```
bash$ scp -P3022 t1.txt devel2@localhost:
(devel2@localhost) Password for devel2@netbsd:
t1.txt                              100%   39    17.1KB/s   00:00
```

Soll ein vollständiges Verzeichnis kopiert werden, muss die Option `-r` angegeben werden und die Quellverzeichnisangabe sollte nicht über ein abschließendes `/` verfügen.

## 2.7 Betrachten und Bearbeiten von Dateien

Mittels des Kommandos `file` lässt sich herausfinden, welches Dateiformat eine Datei hat. Dies kann hilfreich sein, um geeignete Programme zu identifizieren, mit denen sich eine Datei betrachten oder bearbeiten lässt.
Dateityp anzeigen:

file

```
bash$ file hello.txt
hello.txt: ASCII text
```

Auf einigen Linux-Systemen ist das Programm `xdg-open` installiert, welches für angegebene Dateitypen das passende Betrachtungs- bzw. Bearbeitungsprogramm startet (dies entspricht einem Doppelklick auf eine Datei in grafischen Oberflächen). So wird von `xdg-open manual.pdf` ein installierter PDF-Betrachter gestartet. Falls `xdg-open` nicht zur Verfügung steht, muss das jeweilige Betrachtungs- bzw. Bearbeitungsprogramm von Hand gestartet werden.
Es existieren viele Editoren für Textdateien:

- **Texteditoren mit GUI:** gedit, featherpad, Kate Pluma
- **Texteditoren mit TUI:** micro, nano, joe, vim, emacs
- **Texteditoren mit CLI:** In der UNIX-Welt ist Textmanipulation mit auf der Kommandozeile weit verbreitet, da sich hiermit vieles automatisieren lässt. Weiteres dazu findet sich im Modul Betriebssysteme. Hier ein paar Tools, die dabei zum Einsatz kommen: `grep`, `sort`, `uniq`, `sed`, `awk`, `perl`, `diff`, `patch`.

## 2.8 Bedienung mit Maus und Tastatur

Tastaturen verfügen über Modifiziertasten, mit denen unter anderem Befehle abgesetzt werden können (CTRL, ALT, OPT, etc.). Die von Microsoft bei Windows verwendeten Tastenkombinationen (so genannte Tastenkürzel) werden bei vielen Software-Systemen verwendet[4].
In der folgenden Tabelle sind die wichtigsten Tastenkürzel[5] aufgelistet:

[4] Im Netz werden diese auch gelegentlich als CUA shortcuts bezeichnet (von IBMs Common User Access Spezifikation).

[5] Auf deutschen PC-Tastaturen ist die CTRL-Tast mit STRG beschriftet; auf Macs wird die Command-Taste verwendet.

| Kontext | Tastenkürzel | Wirkung |
|---|---|---|
| CUA | CTRL – S | save |
| CUA | CTRL – Q | quit |
| CUA | CTRL – C | copy selection to clipboard |
| CUA | CTRL – V | paste clipboard |
| CUA | CTRL – F | search |
| CUA | CTRL – Z | undo |
| CUA | CTRL – N | new window |
| CUA | CTRL – T | new tab |
| CUA | CTRL – W | close tab or window |
| Window Manager | ALT – TAB | cycle windows |
| Window Manager | ALT – SHIFT | switch keyboard layout US / German |
| Shell | CTRL – A | go to beginning |
| Shell | CTRL – E | go to end |
| Shell | CURSOR UP | last command |
| Shell | TAB | completion |
| Terminal | SHIFT – CTRL – C | copy |
| Terminal | SHIFT – CTRL – V | paste |
| Terminal | CTRL – C | (kill process) |
| Terminal | CTRL – D | EOF (end of file) |
| Terminal | CTRL – Z | suspend process; bg, fg afterwards |

# 3 Material zum aktiven Lernen

## 3.1 Vertiefungsaufgaben zu ‘REPLs’

a) TBD

## 3.2 Vertiefungsaufgaben zu ‘Dateien und Verzeichnisse’

a) Geben Sie die Kommandos an, um eine Verzeichnishierarchy anzulegen, welche das Jahr 2010 mit seinen vier Quartalen und den Monaten widerspiegelt.

b) Geben Sie das Kommando an, mit dem Sie eine Datei `1.txt` in `1.txt` umbenennen

c) Geben Sie das Kommando an, mit dem Sie eine Datei `1.txt` zu `2.txt` kopieren

d) Geben Sie das Kommando an, mit dem Sie eine Datei `1.txt` aus dem Heimverzeichnis in den Ordner `tmp/`, der ebenfalls im Heimverzeichnis ist, verschieben. Das Kommando soll in beliebigen Arbeitsverzeichnissen funktionieren

e) Gegeben ist die Verzeichnishierarchie `Studium/EI/`, `Studium/Mathe1/uebung1/`, `Studium/Mathe1/uebung2/` (alles im Heimverzeichnis). Es soll die Datei `ue2.txt` vom Verzeichnis `uebung1` in `uebung2` verschoben werden. Geben Sie das Kommando dazu an. Das Arbeitsverzeichnis `/tmp/` soll dabei nicht verlassen werden

f) Geben Sie das Kommando an, mit dem Sie alle Dateien, die auf `.txt` enden, finden: - nur die im Heimverzeichnis - im Heimverzeichnis und allen darunterliegenden Verzeichnissen

g) Geben Sie das Kommando an, mit dem Sie alle Dateien, die das Wort `Mathematik` enthalten, finden

h) Es gibt eine Datei 'wichtig.xls' sowie mehrere Kopien davon: `wichtig-1.xls`, `wichtig-2.xls`, `wichtig-3.xls`. Geben Sie das Kommando an, mit dem die drei Kopien aufgelistet werden

i) Geben Sie das Kommando an, um eine Verzeichnis ('Studium') in einer `tar`-Datei (tarball) zu sichern

j) Richten Sie für ihre Linux-VM eine Weiterleitung des TCP Port `2022` auf `22` des Gastes ein, um vom Wirt-OS per `ssh` und `scp` zugreifen zu können.

k) Melden Sie sich per `ssh` in ihrer Linux-VM an.

l) Kopieren Sie eine Datei vom Wirt-OS in ihre Linux-VM per `scp`.

m) Kopieren Sie eine Datei vom Wirt-OS in ihre Linux-VM per `scp` über einen dritten Rechner.

## 3.3 Vertiefungsaufgaben zu 'Betrachten und Bearbeiten von Dateien'

a) Starten Sie ihren GUI-Editor aus dem Terminal heraus (mit und ohne Dateiname).

## 3.4 Vertiefungsaufgaben zu 'Tastenkürzeln'

a) Kopieren Sie Text aus einer Textdatei in eine andere Textdatei.

b) Kopieren Sie Text aus dem Terminal in eine Textdatei.

c) Kopieren Sie Text aus einer Textdatei in ein Terminal.

## 3.5 Vertiefungsaufgaben zu Zusammenfassung

a) Erstellen Sie eine Zusammenfassung des Inhalts dieses Materialpakets (Spickzettel, cheat sheet).

## 3.6 Verständnisfragen zu 'REPLs'

a) Diskutieren Sie, ob es weitere REPL-artige Systeme gibt, als die hier vorgestellten.

### 3.7 Verständnisfragen zu 'Dateien und Verzeichnisse'

a) Welche Vorteile hat es, dass es den Mechanismus „working directory“ gibt?

b) Welche Vorteile hat es, dass es den Mechanismus „home directory“ gibt?

c) Welche Vorteile haben Verzeichnishierarchien?

d) Welche Nachteile haben Verzeichnishierarchien?

### 3.8 Verständnisfragen zu 'Betrachten und Bearbeiten von Dateien'

a) Warum sollten Sie auch wenigstens einen TUI-Editor bedienen können?

b) Welche Vorteile haben GUI und TUI-Editoren gegenüber solchen auf Smartphones?

### 3.9 Verständnisfragen zu 'Tastenkürzeln'

a) Warum ist es wichtig, die Tastenkürzel zu beherrschen?

b) Warum muss im Terminal zu CTRL – C noch SHIFT gedrückt werden?

### 3.10 Testate zu 'REPL'

a) Verwenden Sie eine REPL ihrer Wahl, um einfache Berechnungen durchzuführen (der Art `27290360 / 1024 / 1024`).

### 3.11 Testate zu 'Files'

a) Laden Sie die Datei `RgbMosaik.tar`[6] in ein neues leeres Verzeichnis. Lassen Sie den Inhalt anzeigen und entpacken den Inhalt. Öffnen Sie von der Kommandozeile aus die Datei `mosaic.html` in einem Editor sowie in einem Browser. Ändern Sie den HTML-Quelltext leicht und zeigen die Wirkung im Browser. Verschieben Sie anschließend das neue Verzeichnis auf den Desktop.

### 3.12 Testate zu 'Bedienung mit Maus und Tastatur'

a) Verwenden Sie einen Editor ihrer Wahl und bearbeiten einen Text ausschließlich mit der Tastatur.

## 4 Anhang: Literatur und weiterführendes Material

**Bücher und Papers:**

- William Stallings: Operating Systems: Internal and Design Principles
- Andrew S. Tanenbaum: Modern Operating Systems

[6] `https://www.technik-emden.de/~clink/RgbMosaik.tar`

- Silberschatz, Galvin, Gagne: Operating System Concepts
- The Art of Unix Programming, Eric Steven Raymond[7]
- M. Stonebank – UNIX Tutorial for Beginners[8]

**nützliche URLs:**

- tldr pages – Simplified and community-driven man pages[9]
- Manned.org – aims to index all manual pages from a variety of systems, both old and new, and provides a convenient interface for looking up and viewing the various versions of each man page.[10]
- JSLinux – Run Linux or other Operating Systems in your browser![11]
- https://docs.oracle.com/en/java/javase/22/jshell/introduction-jshell.htm-lOracle – Java Shell User's Guide
- micro – a modern and intuitive terminal-based text editor[12]
- Microsoft, Keyboard shortcuts in Windows[13]
- Ubuntu MATE keyboard shortcuts[14]
- KDE Common Keyboard Shortcuts[15]

**Fundgrube:**

- Terminal Trove – List of Terminal Tools (A-Z)[16]
- ACME – a powerful text editor, development environment and textual-user-interface platform developed by Rob Pike originally for Plan 9 from Bell Labs research operating system[17]

[7] http://www.catb.org/~esr/writings/taoup/html/
[8] http://www.ee.surrey.ac.uk/Teaching/Unix/
[9] https://tldr.sh
[10] https://manned.org
[11] https://bellard.org/jslinux/
[12] https://micro-editor.github.io/
[13] https://support.microsoft.com/en-us/windows/keyboard-shortcuts-in-windows-dcc61a57-8ff0-cffe-9796-cb9706c75eec
[14] https://guide.ubuntu-mate.org/#page-shortcuts
[15] https://docs.kde.org/stable5/en/khelpcenter/fundamentals/kbd.html
[16] https://terminaltrove.com/list/
[17] http://acme.cat-v.org